Integración y Visualización de Datos Abiertos Medioambientales

PAVEL LLAMOCCA PORTELA

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA, FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Máster en Ingeniería de Computadores
Madrid, Junio de 2016

Directora: Dra. Victoria López López

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Máster en Ingeniería de Computadores
Convocatoria: Junio de 2016
Título: Integración y Visualización de Datos Abiertos Medioambientales
Autor: Pavel Llamocca Portela
Directora: M.Victoria López López

Calificación: Apto (5.75)

Firma:

Autorización de difusión y utilización

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster titulado "Integración y Visualización de Datos Abiertos Medioambientales", realizado durante el curso académico 2015-2016 bajo la dirección de doña Victoria López López del departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su reservación y acceso a largo plazo.

Pavél Llamocca Portella.

Madrid, 20 de Junio del 2016

Dedicatoria

Dedico este trabajo a:

Mis padres Daniel y Violeta.

Por haberme apoyado en todo momento, por sus sabios consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor. Por los ejemplos de perseverancia y constancia que me inculcaron siempre sin escatimar ningún tipo de esfuerzo. Por el valor mostrado para salir adelante en momentos complicados.

Mis hermanos Daniel, Patricia y Jeanette

Por todas las enseñanzas recibidas y por mostrarse siempre predispuestos a ser un soporte emocional en mi vida a pesar de los miles de kilómetros que nos distancian.

A mi maestra Victoria

Por ser la guía para la culminación de este trabajo, por el tiempo dedicado y por valorar mucho el trabajo que venimos realizando.

Agradecimientos

Quiero agradecer a todas aquellas personas que me han animado a llevar esta maestría, ya que han sido infinitas las experiencias y habilidades que he adquirido desde su inicio. También agradecer a mi familia por siempre darme esas palabras de ánimo en los momentos que más las he necesitado.

Pero de manera muy especial, a mi maestra, tutora y guía Victoria, por la dedicación que le ha dado a este trabajo, por las palabras de aliento que recibí, por la paciencia que ha tenido que sacar. Pero sobre todo, por la comprensión y el apoyo recibido por su parte, demostrándome que más allá de una profesora, puedo encontrar en ella una excelente persona de la cual estaré eternamente agradecido.

Índice

Índice de Figuras

Índice de Tablas

Resumen

En la actualidad, muchos gobiernos están publicando (o tienen la intención de publicar en breve) miles de conjuntos de datos para que personas y organizaciones los puedan utilizar. Como consecuencia, la cantidad de aplicaciones basadas en Open Data está incrementándose. Sin embargo cada gobierno tiene sus propios procedimientos para publicar sus datos, y esto causa una variedad de formatos dado que no existe un estándar internacional para especificar los formatos de estos datos. El objetivo principal de este trabajo es un análisis comparativo de datos ambientales en bases de datos abiertas (Open Data) pertenecientes a distintos gobiernos. Debido a esta variedad de formatos, debemos construir un proceso de integración de datos que sea capaz de unir todos los tipos de formatos. El trabajo implica un pre-procesado, limpieza e integración de las diferentes fuentes de datos.

Existen muchas aplicaciones desarrolladas para dar soporte en el proceso de integración por ejemplo Data Tamer, Data Wrangler como se explica en este documento. El problema con estas aplicaciones es que necesitan la interacción del usuario como parte fundamental del proceso de integración. En este trabajo tratamos de evitar la supervisión humana aprovechando las similitudes de los datasets procedentes de igual área que en nuestro caso se aplica al área de medioambiente. De esta forma los procesos pueden ser automatizados con una programación adecuada.

Para conseguirlo, la idea principal de este trabajo es construir procesos ad hoc adaptados a las fuentes de cada gobierno para conseguir una integración automática. Concretamente este trabajo se enfoca en datos ambientales como lo son la temperatura, consumo de energía, calidad de aire, radiación solar, velocidad del viento, etc. Desde hace dos años el gobierno de Madrid está publicando sus datos relativos a indicadores ambientales en tiempo real. Del mismo modo, otros gobiernos han publicado conjuntos de datos Open Data relativos al medio ambiente (como Andalucía o Bilbao), pero todos estos datos tienen diferentes formatos. En este trabajo se presenta una solución capaz de integrar todas ellos que además permite al usuario visualizar y hacer análisis sobre los datos en tiempo real. Una vez que el proceso de integración está realizado, todos los datos de cada gobierno poseen el mismo formato y se pueden lanzar procesos de

análisis de una manera más computacional.

Este trabajo tiene tres partes fundamentales: 1. Estudio de los entornos Open Data y la literatura al respecto; 2. Desarrollo de un proceso de integración y 3. Desarrollo de una Interface Gráfica y Analítica. Aunque en una primera fase se implementaron los procesos de integración mediante Java y Oracle y la Interface Gráfica con Java (jsp), en una fase posterior se realizó toda la implementación con lenguaje R y la interface gráfica mediante sus librerías, principalmente con *Shiny*. El resultado es una aplicación que provee de un conjunto de Datos Ambientales Integrados en Tiempo Real respecto a dos gobiernos muy diferentes en España, disponible para cualquier desarrollador que desee construir sus propias aplicaciones.

Palabras clave: Open Data, Medioambiente, Lenguaje R, Integración de Datos, Visualización, Shiny.

Abstract

Today, many governments are publishing (or intend to publish soon) thousands of data sets for individuals and organizations can use them. As a result, the number of applications based on Open Data is increasing. However each government has its own procedures for publishing their data and this causes important problems while combining a great variety of formats from different sources due to there is still no an international normalization. The main objective of this paper is a comparative analysis of environmental data in open databases (Open Data) from different governments. Because of this variety of formats, we must build a data integration process that is capable of uniting all types of formats. The work involves a pre-processing, cleaning and integration of different data sources.

There are many applications developed to support the integration process such Tamer Data, Data Wrangler as explained in this document. The problem with these applications is that they require user interaction as a fundamental part of the integration process. In this work we try to avoid human supervision taking advantage of the similarities of the datasets from the same area which in our case is applied to the area of environment. Thus the processes can be automated with appropriate programming.

To achieve this, the main idea of this work is to build ad hoc processes adapted to each government sources for automatic integration. This paper specifically focuses on environmental data such as temperature, energy consumption, air quality, solar radiation, wind speed, etc. For two years the Madrid government is publishing its data on environmental indicators in real time. Similarly, other governments have published Open Data datasets relating to the environment (such as Andalusia and Bilbao), but all these data have different formats. This paper presents a solution able to integrate all of them and also that allows the user to view and perform analysis on data in real time. Once the integration process is completed, all data will have the same format and then analysis process can be run in a more computational way.

This work has three main parts: 1. Study of Open Data environment and literature about that; 2. Development of an integration process and 3. Development of Graphical and Analytical Interface. Although integration processes through Java and Oracle and Java Graphical Interface (jsp) were implemented at a later stage in a first phase the entire implementation was done with R language and graphical interface through its

libraries, mainly with Shiny. The result is an application that provides a set of Integrated Environmental Data in Real Time on two different governments in Spain, available to any developer who wants to build their own applications.

Keywords: Open Data, Environment, R Programming, Data Integration, Visualization, Shiny.

Capítulo 1. Introducción y Estado del Arte

A día de hoy una de las tendencias de muchos gobiernos a nivel mundial dentro del concepto de Open Government es la de hacer disponibles los datos recolectados con carácter público para que el ciudadano pueda acceder a ellos de forma totalmente libre [1]. El motivo que lleva a los gobiernos a realizar esto es una intención de ofrecer transparencia sobre sus procedimientos administrativos ante el ciudadano [2]. Existe incluso cierto compromiso por parte de los gobiernos en hacer públicos los datos que han sido recolectados por instituciones estatales, ya que dichas instituciones subsisten gracias al aporte de los ciudadanos mediante el pago de tributos e impuestos [3]. Es así pues que el ciudadano común puede usar estos datos sin preocupación de estar transgrediendo algún derecho de Copyright. Es más, el ciudadano puede sentirse en la libertad de usar estos datos para crear servicios/aplicaciones que puedan ser de uso general para el resto de ciudadanos siempre y cuando estos servicios y aplicaciones sean también gratuitos, aplicando el concepto de reutilización de los datos.

Uno de los pilares del Open Data es darle un valor y utilidad a los datos que ya nuestros gobiernos nos proporcionan [4], sin embargo, dado que son miles los datasets que pueden estar disponibles conteniendo diferente información, nace la imperiosa necesidad de implementar soluciones con las que se pueda aprovechar esos datos, recopilarlos e integrarlos con el fin de crear servicios y aplicaciones.

En España, aunque el concepto de Open Data está en una etapa de inicio [5][6] se está extendiendo cada día más, es así que muchas Comunidades Autónomas están publicando algunos de los datos contables que ya tienen digitalizados. La índole de la información publicada por las instituciones encargadas de publicar los datasets es demasiado vasta, pero en base a la cual se puede correlacionar y obtener muchos análisis y conclusiones interesantes. Por ejemplo, podemos obtener los datos de los accidentes de tránsito en los últimos años, y por otra parte podemos tener disponible la cantidad de lluvias que hubo en los últimos años, y en base a esta información podemos entablar una relación entre ambos indicadores y sobre el cual se puedan deducir conclusiones bastante interesantes. Sin embargo, para poder realizar este análisis, debemos tener en cuenta que es muy probable que ambos datos puedan venir de fuentes distintas y por tanto tengan un formato también diferente. Esto implica la necesidad de implementar un proceso de integración, que puede llegar a ser complejo.

Un detalle importante a destacar es el hecho de que aún no se han definido estándares

internacionales para los datasets que cada gobierno desee publicar [7], este es uno de los puntos pendientes y muy importantes en lo que a este sector se refiere. Aún si existiesen estos estándares, el proceso de integración no se puede evitar, pero la complejidad se reduciría considerablemente.

Uno de los propósitos de este trabajo es mostrar un procedimiento mediante el cual se pueden recopilar e integrar diferentes conjuntos de datos abiertos relativos a una misma temática, reduciendo el trabajo de supervisión de un experto. En este trabajo se ha escogido el tema de datos medioambientales dentro de España para realizar las pruebas. El otro objetivo de este trabajo, y no menos importante es crear una aplicación en base a estos datos una vez que hayan sido recolectados e integrados que permita el uso automatizado de los dataset disponibles.

Como paso preliminar, en este capítulo se analizan a continuación los conceptos relacionados con las políticas de datos abiertos y gobierno abierto. Además se muestran diversas iniciativas al respecto. Para completar el estado del arte, se analizan propuestas de integración de datos existentes en la literatura.

1.1 Open Data y Datos Públicos

El concepto de Open Data se puede entender como una filosofía (iniciada en los años 2004 por la organización Open Knowledge Foundation [8]) mediante la cual una administración pública o un gobierno, ponen los datos que gestionan a disposición de la sociedad. Se basa en la creencia de que los datos deben ser libres para usar y publicar sin limitaciones o controles. Sin embargo, para conseguir este enlace entre gobierno y sociedad existen unos procedimientos que tienen que ser implementados para que los datos del gobierno sean de fácil uso y lectura para la sociedad. La idea del Open Data involucra al gobierno haciéndole responsable de hacer disponibles los datos que gestiona, sin embargo, el gobierno puede o no realizar cierta transformación de los datos antes de publicarlos lo que puede facilitar o no su uso posterior.

Estos procedimientos que se encuentran entre los datos de un gobierno y la sociedad, son aquellos que generan servicios a la sociedad en base a los datos públicos y pueden ser implementados por los sectores privado y público o por personas en particular. Son estos servicios los que hacen más fácil a la sociedad el uso de estos datos. Pero para mantener el concepto de Open Data, estos servicios tienen que ser también de uso abierto.

Tomando estas premisas y de acuerdo con [9], un conjunto de datos se puede considerar

Open Data si cumple los siguientes principios:

1. **Los datos son públicos**: Se deben abrir todos los datos de carácter público (todos aquellos, claro está, que no estén sujetos a restricciones de privacidad, seguridad o derechos de autor). Así, no debería existir ninguna otra traba por parte de la administración a la hora de decidir qué datos es pertinente publicar.
2. **Los datos están suficientemente detallados:** Hay que publicar los datos tal y como están en su origen, sin procesar y, por tanto, manteniendo el mayor nivel de detalle posible, lo que se conoce como datos en bruto.
3. **Los datos están actualizados:** Los datos deben ser puestos a disposición de los usuarios con la frecuencia necesaria para que no pierdan su valor y sean precisos y actuales.
4. **Los datos son accesibles:** Hay que hacer accesibles los datos al mayor número de usuarios posible. No debería existir ninguna restricción para todos aquellos que quieran hacer uso de los datos, ni en el propósito de uso.
5. **Los procesos para descargar los datos están automatizados:** Los datos deben estar estructurados para que puedan ser procesados de forma automática por un ordenador. Esta es una condición muy importante ya que el concepto de reutilización de los datos es la razón de ser de la filosofía Open Data y la automatización facilita la reutilización en lo que se viene a denominar la 'democratización de los datos' [10].
6. **Los datos deben ser accesibles sin registro:** En la misma línea que el punto anterior, los datos deben estar disponibles para todos, sin necesidad de identificarse previamente.
7. **Los datos deben ser abiertos:** Los formatos de los datos deben ser no propietarios, es decir, no pueden depender de una entidad o de una herramienta propietaria de una entidad. Como ejemplo, un formato abierto sería csv o xml, mientras que formatos propietarios serían Word, Excel, etc.
8. **Los datos deben ser libres:** Los datos deben ser de uso 100% libre para los usuarios. Así, los datos deben estar libres de derechos, patentes, copyright y no estar sujetos a derechos de privacidad, seguridad o privilegios que puedan estar reglados por otras normas.

Existen instituciones que poseen diversos tipos de datos sin embargo, no todos son publicables. Por lo general, el sector estatal es donde se publican más datos como

consecuencia de la aplicación de políticas de transparencia, ya que existe un fuerte compromiso por parte de los gobiernos en hacer de conocimiento público los datos recabados con dinero público y con los que se está gestionando un país, una región o una ciudad.

Los datos públicos dentro del ámbito de Open Data deben ser aquellos en base a los cuales se puedan generar servicios. Por ejemplo: transporte, ciencia, productos, educación, sostenibilidad, mapas, legislación, bibliotecas, medio ambiente, economía, cultura, negocios, diseños, finanzas, etc.

Según Laura James, CEO de la Open Knowledge Foundation, existen dos elementos para que los datos se consideren públicos dentro del concepto de Open Data [11]:

- **Apertura Legal**: Debe estar permitido legalmente obtener los datos, trabajarlos y compartirlos. Usualmente estos datos tienen una "open licence" que permite el libre acceso, reutilización de los datos y su publicación en un dominio público.
- **Apertura Técnica**: No deben existir barreras técnicas para usar los datos. Por ejemplo, publicar los datos en tablas o documentos complican el uso de estos datos. Los datos deben ser de fácil procesamiento para un ordenador, lo que se conoce como "machine readable", es decir, deben ser publicados en ficheros planos, por ejemplo el formato ya mayoritariamente aceptado csv.

1.2 Open Government

La característica principal de los "Gobiernos Abiertos" es la transparencia. En los últimos años la ciudadanía y los gobiernos están muy concienciados en la promoción de políticas de transparencia, muchas veces en demanda de una recuperación de la confianza de los ciudadanos en los gobernantes y las instituciones públicas. Para conseguir esto, un gobierno debe contar con las plataformas necesarias con las que pueda hacer transparentes toda su información publicable (a excepción de datos sensibles según las legislaciones aplicables en cada caso, por ejemplo datos de carácter personal, compromiso de la seguridad de la ciudadanía, etc.). En los últimos años la transparencia se ha mostrado como un sello de calidad de un gobierno, pero esto implica que cualquier persona podrá tener acceso a la información del gobierno y ser informado de los procedimientos del mismo. En los últimos años el concepto de Open Government ha incluido otros factores como es la participación y colaboración ciudadana como parte de la implementación de sus procedimientos en el uso de la tecnología.

1.2.1 Open Government Data

Los datos de gobierno abierto hacen referencia a los datos producidos o comisionados por un gobierno o entidades controladas por el gobierno y que al mismo tiempo estos datos puedan ser usados, reutilizados y re-distribuidos por cualquier ciudadano de manera totalmente libre.

Existen muchas razones por las cuales un gobierno debería abrir sus datos, las tres más importantes son:

Transparencia. En un gobierno democrático y de buen funcionamiento, los ciudadanos necesitan conocer qué es lo que está haciendo el gobierno. Para esto, ellos deben ser capaces de acceder libremente a la información del gubernamental y poder compartirla con otros ciudadanos. La idea de transparencia no trata solo del acceso, sino también de reutilizar y compartir.

Brindar valor comercial y social. En un era digital, los datos son una fuente de información para actividades sociales y comerciales. Cualquier tipo de búsqueda necesita el acceso a los datos, muchos de los cuales son creados o están en posesión del gobierno. Al publicar los datos, los gobiernos pueden ayudar en la creación de negocios innovadores y servicios que ofrecen valor social y comercial.

Gobierno de participación. Habitualmente los ciudadanos están comprometidos con su gobierno muy esporádicamente, a veces únicamente cada 4 años con motivo de elecciones. Sin embargo, en los últimos años ha crecido el interés por concienciar sobre la participación ciudadana en el gobierno de las ciudades, entre otros. Muchas páginas web de ayuntamientos contienen una llamada para la participación, por ejemplo mediante la solicitud de completar formularios donde el ciudadano puede expresar opiniones o realizar propuestas. El ayuntamiento de Madrid (www.madrid.org) es un ejemplo de ello. Al publicar los datos, los ciudadanos estarán directamente más informados y el uso de los datos deriva en una participación más cercana de los ciudadanos en la toma de decisiones del gobierno, al conocerse los procedimientos del gobierno y propiciar que los ciudadanos sean capaces de contribuir con sus propuestas o críticas.

Sin embargo, es muy importante mencionar que para los datos gubernamentales suelen existir ciertas restricciones sobre la información a publicar. Se tiene que considerar que

existen ciertos tipos de datos que por restricciones nacionales de seguridad no son publicables y sobre los cuales los gobiernos trabajan internamente. También cabe mencionar que no pueden publicarse datos de carácter personal (DCP) protegidos formalmente por las leyes de cada país.

1.3 Iniciativas Open Data

La idea de Open Data empezó a generar atracción algunos años atrás, y cada año se van generando nuevas iniciativas alrededor de todo el mundo respecto a este tema. Sin embargo, después de muchos años de esfuerzo, aún seguimos considerando al Open Data como un concepto nuevo. A pesar de que es una idea que ha venido ganando una importante relevancia política, su potencial e implicación para los gobiernos está empezando a articularse y la evidencia de su impacto está empezando a emerger. Se ha realizado un estudio en el año 2013 sobre las iniciativas en Open Data en el espacio EMEA [12] y algunas conclusiones importantes pueden ser:

- Solo un 7% de los datasets publicados cumplen los dos elementos que Laura James menciona: “machine readable” y “open license” [11].
- Los países con un alto índice de innovación en Open Data, se enfocan e invierten más en infraestructura de datos, estructuras de almacenamiento y compromiso de la comunidad.
- Los países con un índice medio de innovación en Open Data, fallan en proveer de un bueno número de datasets valorables y presentan algunas debilidades en las políticas de Open Data como acceso a la información o protección de datos.
- Los países con un bajo índice de innovación, no han empezado sus iniciativas en Open Data aún (lo cual sigue siendo cierto en 2016).

Figura 1. Barómetro de iniciativas de Open Government
(Fuente: The Open Data Barometer 2013 Global Report)

La Figura 1 muestra un barómetro de Open Data por países en el cual los países con colores más oscuros tienen más y mejores iniciativas en comparación con los colores claros. Se observa claramente la relación de Open Data en países con más nivel de desarrollo y la relación entre el índice de innovación y el grado de calidad de los datasets publicados.

Existen muchas iniciativas por las cuales se está tratando de mejorar la publicación y el acceso a los Open Data. Vamos a mencionar algunas distinguiendo entre iniciativas tecnológicas y otras colaborativas entre varios gobiernos o instituciones.

1.3.1 Iniciativas Tecnológicas

En esta sección se describen las iniciativas tecnológicas que se han estudiado previamente a la realización de este trabajo.

Comprehensive Knowledge Archive Network (CKAN)

CKAN [13] fue desarrollado en el año 2007 por la Open Knowledge Foundation, se describe como una plataforma que permite la organización, la publicación y compartición de los datos. Es una herramienta open-source que incorpora productos como SolR y Ubuntu.

CKAN puede ser usado por cualquier institución y cuenta con procedimientos propios que sugieren a las instituciones para cumplir correctamente su cometido. CKAN es una tecnología madura y ampliamente utilizada, por ejemplo el portal de datos del Reino Unido (data.gob.uk) lo utiliza para la publicación de sus datos.

W3C Data Activity - SPARQL

W3C Data Activity [14] es un concepto que nació en el 2013 y combina dos conceptos ya conocidos: Web Semántica y Gobierno Electrónico. El objetivo principal es simplificar lo máximo posible el proceso de compartir los datos, usando la Web como un medio de intercambio de datos de la misma manera que ya sirve para la publicación de documentos. Una explicación bastante práctica sobre búsquedas semánticas, podemos encontrarla en [14].

Aunque esta tecnología no es propia de las iniciativas Open Data, se usa bastante para la búsqueda de datasets. Sparql es un lenguaje de consultas semánticas y existen ya instituciones que tienen implementada alguna plataforma en Sparql que permite obtener datos desde catálogos que ya están disponibles en formato de Linked Data. Con Linked Data entendemos un conjunto de datos que vienen también representados en formato RDF (Resource Description Framework), esto significa que los contenidos están internamente representados en mapas de grafos y permiten realizar una búsqueda semántica. Para usar Sparql se tiene que asumir que el conjunto de información sobre el cual se va a realizar una búsqueda cumple el requisito de Linked Data.

El acceso a los datos se realiza mediante Sparql Endpoints [15] que son terminales en la web en la que se pueden encontrar buscadores Sparql para un tema en concreto. El lenguaje Sparql es bastante completo y permite realizar consultas muy concretas y complejas y los resultados pueden ser devueltos en formato XML, JSON, RDF, HTML.

Este tipo de tecnología se usa para buscar diversos datasets, siendo por tanto también una tecnología madura. Por ejemplo, esta implementado en el portal de datos de Estados Unidos (data.gov) y en el del Reino Unido (data.gov.uk).

Open Government Platform (OGPL)

OGPL [16] es una plataforma Open Source creada en colaboración entre la India y los Estados Unidos para promover la transparencia e incrementar la participación del ciudadano en crear más Open Government Data, documentos, herramientas y procesos públicamente disponibles. Sus principales funcionalidades y características son:

1. Publicar datos de gobiernos, documentos y procesos desde múltiples departamentos dentro del Gobierno.
2. Proporciona un procedimiento para la aprobación, publicación y gestión de datos.
3. Utiliza las tecnologías Open-Source de la Web 2.0 para desarrollar infraestructuras de bajo coste y “cloud-based”.

4. Compromete a los ciudadanos al uso de las aplicaciones y servicios basadas en Open Data para mejorar la calidad de vida.
5. Intenta crear una comunidad en base a tópicos específicos, de prioridad e interés nacional.
6. Permite a los usuarios finales compartir los datos por redes sociales como Facebook o Twitter.
7. Provee API's públicas y otras herramientas para la visualización de datos

1.3.2 Iniciativas no tecnológicas colaborativas

Existen alianzas globales que en algunos casos se enfocan en la perspectiva del gobierno, en otros casos en la perspectiva de la sociedad civil y en otros casos combinan ambas. A continuación describimos algunas de ellas.

Open Government Partnership (OGP)

OGP [17] es un esfuerzo global para dar mayor calidad a la gestión gubernamental que se lanzó en Septiembre del 2011. Esta alianza empezó con 8 países en sus dos primeros años (Brasil, Indonesia, México, Noruega, Filipinas, Sur África, Reino Unido y los Estados Unidos) pero a día de hoy ya son 60 los miembros. Uno de los fines de esta alianza es el de buscar un mejor desempeño del gobierno y de dar una respuesta de calidad al ciudadano, así como también involucrarlo e informarlo de la gestión del gobierno.

Para ser parte de esta alianza, un país debe ser elegido y sus autoridades deben firmar un documento en el que se establezca el compromiso de inversión y focalización en lo que a Open Government se refiere. OGP es supervisado por un Comité Directivo integrado por un número igual de representantes de gobiernos y sociedad civil, este comité está supervisando y auditando constantemente los avances de cada uno de los miembros de la alianza.

International Open Data Charter (IODC)

IODC [18] surgió como resultado de una mejora de lo que se conocía como "Open Data Charter". El "Open Data Charter" fue un acuerdo que firmaron los líderes del grupo G8 en Junio del 2013 y que especificaba 5 principios estratégicos de la base de Open Data sobre los cuales los miembros del G8 deberían actuar. A partir de entonces muchos otros gobiernos se han ido interesando por lo que ha surgido una necesidad de implementar mejoras que cubran unas necesidades globales.

En los meses siguientes algunos grupos multinacionales empezaron a realizar sus actividades para establecer principios de Open Data más inclusivos y representativos. Dentro de los grupos que se sumaron a este esfuerzo estaba también el OGP, dando paso al "International Open Data Charter".

Global Open Data Initiative (GODI)

GODI [19] fue anunciada por el Open Institute (http://www.openinstitute.com) en el año 2013. Es una iniciativa liderada por las organizaciones de sociedad civil (Fundar, Open Institute, Open Knowledge Foundation, The Sunligth Foundation y la World Wide Web Foundation) para compartir principios y recursos para gobiernos y sociedades. Está dirigida a desarrollar, encontrar y compartir herramientas, guías y lecciones relativas a políticas que los gobernantes pueden usar para construir iniciativas de Open Data más satisfactorias. Sus objetivos son:

1. Servir como una voz de guía internacional en la perspectiva de la sociedad en aspectos

de Open Data y conectar los grupos más aislados en cada contexto nacional.

2. Incrementar el interés y proveer una visión para los gobiernos en cómo implementar Open Data de una manera sostenible.
3. Incrementar la conciencia y el uso del Open Data.
4. Ampliar la evidencia en Open Data.

Global Open Data Index

El Global Open Data Index [20] es un esfuerzo anual para medir el estado del Open Government Data alrededor del mundo. Es una encuesta diseñada para evaluar la apertura de datasets de Gobiernos específicos. Por medio de esta iniciativa se desea proveer una auditoria civil acerca de cómo los gobiernos actualmente publican sus datos. Los objetivos del Global Open Data index son:

1. Resultados desde la perspectiva del ciudadano y no simplemente en la fiabilidad de las afirmaciones del Gobierno
2. Un grupo de datasets que ofrecen una vista muy amplia en las funciones más importantes de un Gobierno así como también su rendimiento. Y estas pueden ser comparadas consistentemente con otros países.
3. Proveer de mecanismos especializados por tópico, que sirvan para la revisión de la publicación de un dataset. Así se garantiza la fiabilidad del dataset.
4. Comprometer y educar a los ciudadanos a aprender acerca de los datos abierto de su Gobierno y como pueden hacer un mejor uso de ellos.
5. Establece una línea de base y rastrea cambios y tendencias en el mundo de Open Data al mismo tiempo que en los campos que envuelve.

En el primer año de encuesta (2014) fueron 97 países los involucrados para obtener el Global Open Data Index, en el año 2015, esa cifra aumento a 122 países.

1.3.3 Proyectos Globales

Las siguientes iniciativas son proyectos que han sido empezados con el apoyo de varias instituciones internacionales.

Open Data Research Network (ODRN)

La ODRN [21] está coordinada con la Web Foundation y el International Development Research Centre (IRDC) y se creó en el año 2013. Es una red que está orientada a entender como las iniciativas en Open Data pueden ser implementadas satisfactoriamente y escaladas para tener un impacto. Sus principales objetivos son:

1. Conectar investigadores de Open Data a nivel global
2. Crear información conjunto relativa a las investigaciones en las implementaciones e impactos de las iniciativas en Open Data.
3. Proyectos de Investigación en Servidores de Open Data

Hasta la fecha, este proyecto ha ayudado en iniciativas a miles de instituciones que desean implementar tecnologías de Open Data, desde Ministerios hasta organizaciones activistas. Paralelamente a estos objetivos el ORDN también está llevando a cabo el proyecto "Emerging Impacts of Open Data in Developing Countries (ODDC)" que es de ámbito global cuyo fin es el de entender como el Open Data es puesto en uso y contexto en diferentes países a nivel mundial.

Open Data Partnership for Development (ODP4D)

ODP4D [22] es una alianza entre el Banco mundial, el Open Data Institute y la Open Knowledge Foundation, se firmó en el 2013. Su primer fondo fue de $ 1.25 millones y fue donado del World Bank's Development Grant Facility. Fue diseñado para ayudar a los creadores de políticas y ciudadanos en países en desarrollo a entender y explotar los beneficios del Open Data. Oficialmente aún están en una etapa de iniciación y están en la búsqueda de nuevos "partners" que se unan en su proyecto. Sus objetivos son:

1. Dar soporte a países en desarrollo para plantear, ejecutar y lanzar iniciativas de Open Data a nivel nacional.
2. Incrementar la reutilización de Open Data en países en desarrollo creando estándares, guías, redes regionales y demanda de datos.
3. Incrementar la base de evidencia del impacto del Open Data para el desarrollo.

Se tiene pensado implementar estándares y guías para la explotación de los beneficios de la implementación de Open Data, además se ofrecerá capacitación actual y redes de

programas a cada organización [23].

1.4 Integración de datos

La integración de datos significa combinar datos que se encuentran en diferentes fuentes para permitirle al usuario final tener una vista unificada de los mismos para una accesibilidad idónea, que sirva a las necesidades de negocio [24].

El crecimiento del *Big Data* implica el uso de más recursos para las nubes, centros de almacenamientos y también más herramientas de integración de datos y de limpieza de datos (*data cleaning*). Las empresas consumen gran cantidad de tiempo en extraer, filtrar, limpiar datos de diferentes tipos de sistemas y es lo que según el artículo [25] (y la lógica) se debería tratar de simplificar.

Dado que el verdadero valor de los datos se hace más visible cuando éstos están fusionados con otros datos, se hace crítico encarar los retos que conlleva la integración del *Big Data* (BDI por su acrónimo en inglés Big Data Integration). La BDI es diferente a la integración de datos convencionales por 4 motivos [26]:

Primero. El número de fuentes de datos para un mismo tema, se ha incrementado hasta ser de decenas de miles.
Segundo. Las fuentes de datos son dinámicas ya que estos son continuamente generados.
Tercero. Los datos son extremadamente heterogéneos en su estructura, con variaciones considerables incluso para temas similares.
Cuarto. Las fuentes de datos tienen diferencias de calidad como pueden ser errores de precisión, cobertura, datos vacíos, etc.

Según Xin Luna Dong [26], la era del Big Data es la "inevitable consecuencia de la datificación: nuestra habilidad de transformar cada evento y cada interacción en el mundo en un dato digital, y nuestro deseo de analizar y extraer valor de este dato". A día de hoy, las organizaciones necesitan tecnología de integración suficientemente flexible para manejar *Big Data* independientemente de la procedencia de los datos. Por este motivo, las herramientas para la integración del *Big Data* deben ser capaces de trabajar con una amplia variedad de arquitecturas. Elegir el mejor enfoque a un proceso de integración de *Big Data* puede ser complicado para organizaciones que carecen de conocimiento funcional y de buenas prácticas relativas a sus industrias y a las tecnologías de la información. Un estudio [27] refiere que de

acuerdo al 51% de organizaciones, estas deben de asegurarse de que su equipo IT haga todo lo posible para maximizar las habilidades y recursos internamente y no malgastarlos en esfuerzos inútiles. El tener el proceso de integración y los métodos de manejo de datos adecuados pueden ayudar al equipo de IT de una organización a trabajar más eficientemente y dar mejor soporte a las unidades de negocios.

El autor del artículo [28] considera cada uno de las partes de la integración del Big Data como retos que se deben realizar:

La extracción de datos. El manejo del *Big Data* implica el análisis y el procesamiento de una gran cantidad de datos. El proceso de extracción conlleva realizar una transmisión, acceso y entrega de datos desde un amplio rango de recursos y luego realizar la carga de estos datos dentro de una plataforma de *Big Data*. Estos requisitos de analizar transformaciones y extracciones no son limitados sólo a conjuntos de datos convencionales.

Sincronización de datos. Una vez que los datos sean importados en plataformas de *Big Data*, se observa que las copias de datos migrados a partir de una amplia gama de fuentes en diferentes tiempos pueden estar rápidamente desincronizados con el sistema de origen. Esto implica que los datos procedentes de una fuente no están actualizados, en comparación con los datos procedentes de otra fuente. Este es un aspecto a considerar para que los datos mantengan una consistencia.

Limpieza de datos. Dada la variedad de fuentes de información de las cuales se puede extraer información, se ha de tener en cuenta que los formatos en las que estas se encuentran se han creado para un uso genérico, por lo cual, uno de los puntos que un proceso de integración debe considerar es sobre qué datos se va a realizar el análisis, así podemos descartar un volumen de datos considerable para trabajar solo con aquellos que nos interesan.

Transparencia de datos. El manejo del *Big Data* conlleva a que los datos estén disponibles así como también permite el acceso a los usuarios finales que hacen uso de herramientas de Business Intelligence con el propósito de descubrir relaciones. Estas herramientas de Business Intelligence deberán ser capaces de conectar diferentes plataformas de *Big Data* y además de brindar transparencia de los datos. Al mismo tiempo, si el número de consumidores de datos se incrementa, estas herramientas deben ser capaces de dar acceso simultáneo a todos los usuarios.

Muchas de las tareas de limpieza de datos se realizan aún con programación personalizada porque la disponibilidad de herramientas estandarizadas es escasa [29]. Sin

embargo existen en la actualidad algunas iniciativas de investigación en universidades prestigiosas como MIT con *Data Tamer* y Stanford con *Data Wrangler*.

1.4.1 Data Tamer

La calidad y la integración de los datos es un problema crítico para los analistas de datos y se han desarrollado técnicas para encararlo. Una de estas es *Data Tamer* que ayuda en el proceso de la Limpieza de datos y soporta transformaciones reutilizables [31].

Data Tamer [31] es una herramienta elaborada por el MIT y Qatar Computing Research Institute (QCRI) y tiene como finalidad lo que denominan *Data Curation*. El concepto *Data curation* está englobado dentro de lo que se conoce como integración de datos, *Data curation* es el acto de descubrir fuentes de datos, limpiarlos organizarlos semánticamente con otras fuentes de datos locales y eliminar información redundante. Las principales aportaciones del *Data Tamer* frente a otras herramientas que también realizan labores de *Data Curation* [32] son las siguientes:

Escalabilidad mediante automatización. El problema de muchos de los sistemas de integración de datos es que dependen de la intervención del humano. Los nuevos sistemas tendrán que usar nuevos algoritmos en los que la intervención del humano sea solo la necesaria. A pesar de este intento de automatización no supervisada, este es el punto más débil de Data Tamer.

Limpiado de datos. Las fuentes de datos empresariales están inevitablemente "sucias". Por tanto, los nuevos sistemas requieren la mínima intervención del humano.

No orientada a la Programación. Herramientas actuales de ETL, necesitan programadores profesionales para su uso. Los nuevos sistemas no requerirán usuarios con pocos conocimientos de programación para realizar tareas de integración.

Incremental. Nuevas fuentes de datos siempre irán apareciendo, por tanto no se concibe que el proceso de integración se dé por finalizado.

Para el funcionamiento del *Data Tamer* se utilizan dos roles humanos:

Data Tamer Administrator (DTA). Es análogo a lo que es un DBA. Este debe especificar la colección de fuentes de datos para realizar el *Data Curation.*

Domain Expert (DE). Son expertos de dominio que resuelven dudas que surjan durante el

Data Curation.

Cuenta también con una interface gráfica que permite al DTA y al DE realizar sus propias acciones. A esta interface se le denomina *DTA Management Console*, y algunas de las acciones que permite realizar son las siguientes:

Ingesta. Procesar una nueva fuente de datos y almacenarla dentro de una base de datos Postgres.

Identificación de atributos. Asignar a una fuente de datos los atributos necesarios para su posterior clasificación.

Consolidación de entidades. Asignar una categoría a una entidad en base a los atributos asignados. Una entidad se refiere a una categoría por las cuales los datos pueden ser agrupados u organizados.

La Arquitectura del *Data Tamer* se muestra en la Figura 2.

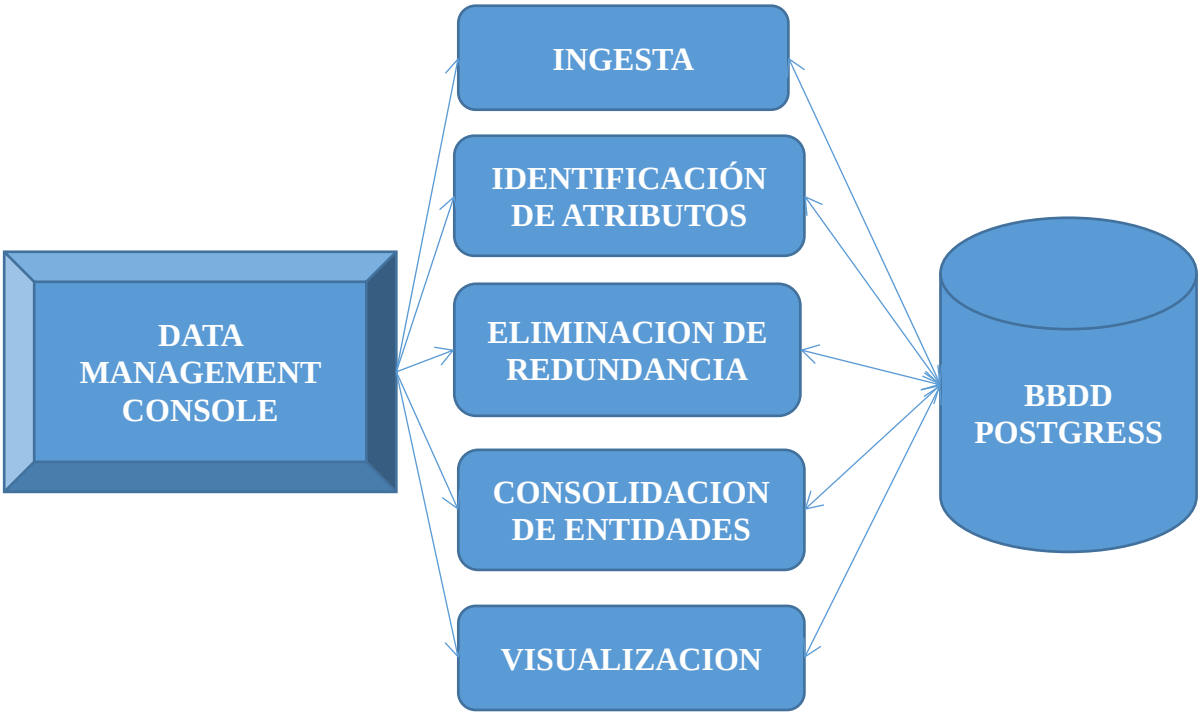


Figura 2. Arquitectura de Data Tamer

Uno de los usos más actuales para los que se usa esta herramienta es para la integración de texto en estructuras de datos ya diseñadas [33], por ejemplo, las nuevas fuentes de información se nutren a través de información que llega por medios electrónicos (Web blogs,

Twitter, Facebook, etc.), y estos datos se actualizan instantáneamente, obligando a las empresas a contar con una cobertura de datos bastante amplia. Es por esto que muchas empresas están interesadas en usar herramientas que permitan integrar texto no-estructurado en sus estructuras de datos. En la Figura 3 se muestra los procesos propios del *Data Tamer* usados dentro de una arquitectura que permita elaborar sistemas de este tipo.

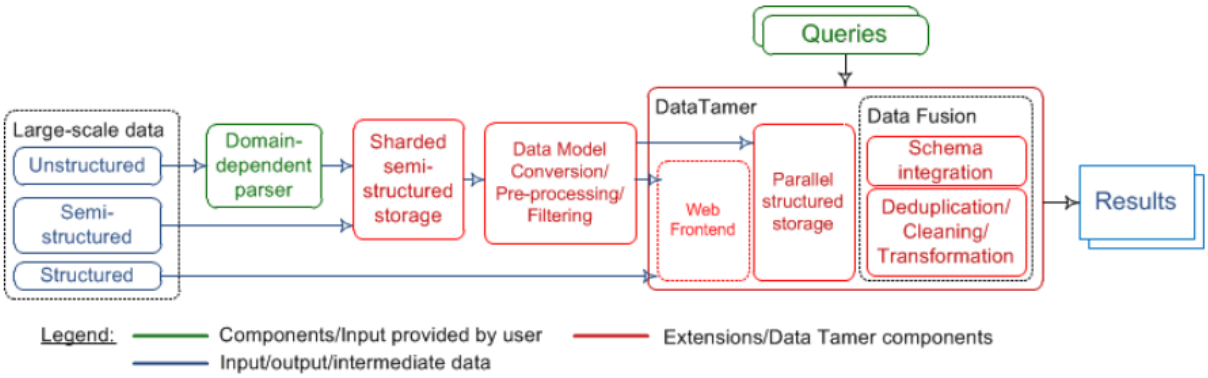


Figura 3. Arquitectura Data Tamer para integración de texto

1.4.2 Data Wrangler

Data Wrangler se refiere a un sistema interactivo para la transformación de datos desarrollado por la universidad de Stanford llamado *Wrangler* [34]. Es una aplicación web escrita en JavaScript. *Wrangler* combina una interface de usuario de iniciativa mixta con un lenguaje de transformación declarativo. *Wrangler* está diseñado en base a un lenguaje de transformación.

Este lenguaje contiene un conjunto de operadores, que incluyen funcionalidades para el limpiado de datos, agrupación, roles semánticos y operadores complejos de formateo. Además este lenguaje cuenta con 8 clases de transformaciones:

Map Transform. Mapea un registro de datos de entrada a cero, uno o múltiples registros de salida.

Delete Transform (one-to-zero). Evalúa predicados para determinar qué filas se eliminarán.

One-to-one Transform. Permite la extracción, corte y separado de valores en columnas múltiples.

One-to-many Transform. Permite operaciones de separado de datos en múltiples registros. Por ejemplo, formatear una línea de texto de un fichero en nuevas líneas.

Lookups Transform. Incorpora datos desde tablas externas. *Wrangler* incluye lookups comunes como por ejemplo del código postal para la agrupación a través de ciudades.

Join Transforms. Actualmente *Wrangler* soporta dos tipos de Join: equi-join y approximate transform. Estos son útiles para generar lookups y para corregir errores de tipografía de datos conocidos.

Reshape Transform. Manipula la estructura de tablas y esquemas. *Wrangler* soporta dos operadores para esta tarea: *Fold* y *Unfold*. Esta transformación permite una restructuración de orden superior y es común en herramientas como R y Excel Pivot Tables.

Positional Transform. Incluyen los operadores *Fill* y *Lag*. Fill genera valores basados en valores vecinos en una fila o columna. El operador Lag desplaza los valores de una columna por arriba o por debajo por una determinada cantidad de líneas.

Además, el lenguaje incluye funciones de ordenación, agrupación (sum, min, max, mean, etc.) y de generación de claves. La interface gráfica del *Wrangler* permite al usuario 6 operaciones básicas dentro de una tabla: seleccionar tablas, seleccionar columnas, clicar barras, seleccionar texto dentro una celda, editar valores dentro de una tabla y asignar nombres de columnas, tipos de datos o roles semánticos.

Otra característica interesante del *Wrangler* es que permite sugerir transformaciones en base a un historial de interacciones para un usuario que se haya registrado. Estas sugerencias se pueden realizar de 3 maneras distintas considerando diferentes criterios. Por último, *Wrangler* cuenta con un proceso de validación de la transformación. Encima de cada columna se encuentra el "data quality meter" que es una barra que indica la proporción de valores que han sido verificados completamente.

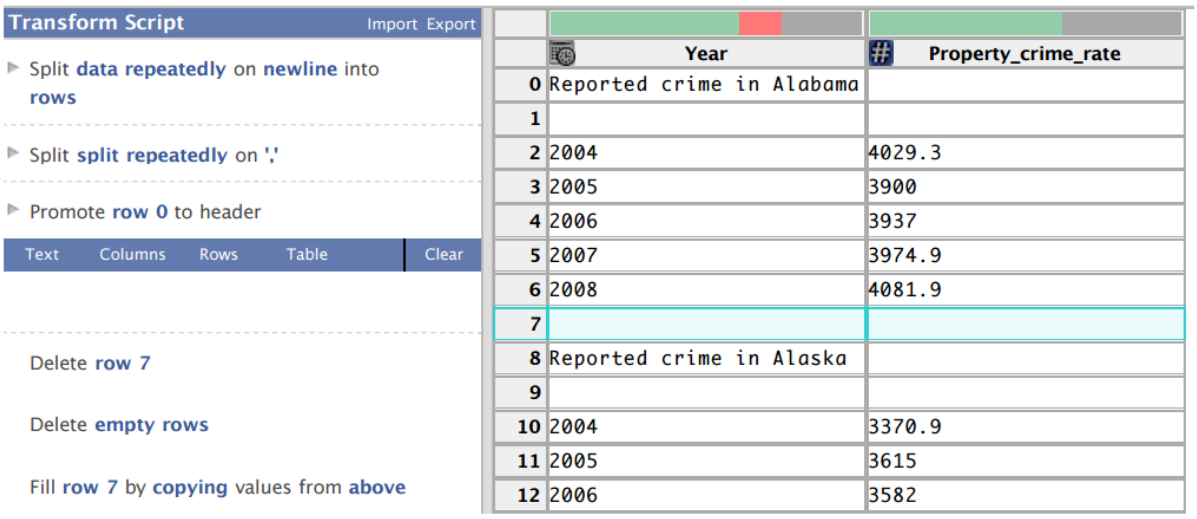


Figura 4. Interface interactiva del Data Wrangler

En la Figura 4 se muestra la interface gráfica del *Wrangler*, el panel de la derecha contiene (de arriba hacia abajo) un historial de transformaciones, un menú de selección de transformación y sugerencias automáticas de transformaciones en base a la selección actual. Los textos en negrita dentro de las descripciones de las transformaciones indican los parámetros que pueden ser usados y revisados. El panel derecho contiene una tabla interactiva, encima de cada columna se encuentra el "*data quality meter*".

Como conclusión de esta sección, podemos destacar que la principal debilidad de las herramientas de integración investigadas es la necesidad de supervisión del proceso sobre todo en los estados iniciales. El problema se ve agravado al ser herramientas orientadas a propósito general y a ser usadas por grandes corporaciones con todo tipo de datos. Nuestra idea será por tanto, discriminar las temáticas para poder hacer una integración de los datos que se pueda automatizar en base al conocimiento que se tiene previo del área donde se aplica. En nuestro caso sobre datos medioambientales.

1.5 Objetivos y Organización de esta Memoria

1.5.1 Cuestiones y objetivos

En este Trabajo de Fin de Máster se desarrollan los temas de interés para dar respuesta a las siguientes cuestiones de investigación.

Cuestión 1. Cómo acceder y descargar (si es necesario) datasets procedentes de diferentes fuentes pero con un interés común de estudio.

Cuestión 2. Cómo integrar los datos en un mismo formato que permita el análisis conjunto de los datasets.

Cuestión 3. Cómo visualizar resultados de análisis en tiempo real de datos procedentes de diversas fuentes y con distintos formatos.

Tras estudios precedentes en la Facultad de Informática de la UCM, (universidad donde se presenta este trabajo) y en colaboración con el Ayuntamiento de Madrid como institución pública facilitadora de los datos abiertos utilizados y con el fin de analizar estas cuestiones, se ha elegido el área de medioambiente como área temática de los datasets. Para dar respuesta a las cuestiones de investigación planteadas, en este trabajo se proponen dos objetivos:

Objetivo 1. Desarrollar una herramienta que sea capaz de acceder a los datos de interés e integrarlos en tiempo real en un dataset común con formato adecuado sin requerir la presencia

de un analista de datos que supervise el proceso, habida cuenta de que con supervisión ya existen varias propuestas sobre integración de datos como se ha comentado en la sección anterior. Estas propuestas, no obstante, han sido analizadas por el autor de este trabajo para comprender sus objetivos, características y funcionalidades. Sin embargo, no pueden utilizarse como solución a las cuestiones planteadas en este trabajo por ser de aplicación general, lo que implica una supervisión continua y muy intensa en la fase inicial de la integración. Con este objetivo el autor trata de dar respuesta a las cuestiones 1 y 2.

Objetivo 2. Proporcionar una plataforma o portal de recuperación, análisis y visualización de datos abiertos e integrados de acuerdo al Objetivo 1. Con este objetivo, el autor trata de dar respuesta a la cuestión 3.

1.5.2 Organización de la memoria

Además de este capítulo introductorio, en el Capítulo 2 se describe ampliamente el desarrollo del fenómeno Open Data en España y se realiza un estudio profundo de los datos medioambientales accesibles. El Capítulo 3 está dedicado al proceso de integración de datos en tiempo real desarrollado por el autor desde dos perspectivas de gobierno: datos procedentes de la Comunidad de Madrid y datos procedentes de la Junta de Andalucía. En ambos casos se realiza unas versiones open source con R y en el primer caso, además se realiza una comparación de rendimiento de esta versión con una versión Java-Oracle desarrollada en una fase previa a modo de prueba. En este capítulo se trata de dar respuesta al Objetivo 1. El Capítulo 4 da respuesta al Objetivo 2. En él se muestran las versiones de la interface gráfica desarrollada para visualizar y analizar los datos integrados con los procesos descritos en el capítulo anterior. Se realizan pruebas sobre los datos medioambientales de la Comunidad de Madrid y de la Junta de Andalucía para mostrar los resultados desarrollados en las diferentes versiones. Finalmente, en el Capítulo 5 se realizan las conclusiones y se propone el trabajo futuro.

Capítulo 2. Open Data en España y Datos Abiertos sobre Medioambiente

En este capítulo se aborda la situación de los datos abiertos en España y concretamente los datos medioambientales. Se ha realizado un estudio a nivel de Europa en relación a las características de los datasets públicos y los resultados para España [35] indican que aún se deben seguir realizando esfuerzos para implementar más iniciativas en el área:

- 77.32 % de los datasets están disponibles en formato de descarga.
- 78.35 % de los datos no están acompañados de una licencia abierta.
- 81.00% de los datasets soportan dos o más lenguajes, el más soportado es el Inglés y pocos casos el Catalán, Español y el Vasco.

Sin embargo, los esfuerzos realizados también son bastante importantes. Por ejemplo el gobierno español ha lanzado un estándar de interoperabilidad técnica para el sector público que especifica instrucciones para entidades públicas que usan Open Data y que marcan un hito importante para la reutilización de los datos desde Administraciones Públicas [36]. Otro aspecto que es importante resaltar, es que en España se han lanzado varios proyectos para las Administraciones Públicas que han sido reconocidos por la Unión Europea como el proyecto "Aporta" cuya finalidad es promover la reutilización de la información del sector público y fomentar una cultura de Open Data entre la administración pública y la sociedad. También es importante mencionar que el gobierno español se ha implicado en las iniciativas de Open Data y ha lanzado un foro de colaboración con el sector privado para la reutilización de información pública. Al mismo tiempo tampoco podemos dejar de mencionar algunos esfuerzos realizados pero que requieren una mayor participación de los instituciones públicas, como lo es la iniciativa de Aenor con la Norma UNE 178301:2015 que intenta crear una normalización para la publicación de datasets por parte del sector público.

Una iniciativa bastante innovadora podemos encontrar en el CTIC (Centro Tecnológico de la Información y la Comunicación) en Asturias. El CTIC [37] cuenta con un registro en el cual va inventariando todos los portales en los que se ofrezca algún Open Data que hay no sólo en territorio español, sino también a nivel global:

Para mantener este registro, el CTIC hace uso de "búsquedas facetadas". Estas búsquedas a diferencia de las demás, devuelven un número reducido de resultados basados en la clasificación que todos los usuarios hacen de un determinado tema. Esta clasificación se hace a través de propiedades, atributos (facetas). Para esto, previamente los temas han tenido

que ser categorizados y estructurados.

Con todo esto tenemos ya centralizadas las ubicaciones de todas aquellas instituciones que ofrecen algún tipo de Open Data. Las características de los conjuntos de datos de los Open Data encontrados son infinitas y muy variables. Podemos encontrar distintos ámbitos, distintas fechas de publicación, distintos formatos, distinta frecuencia de publicación, etc. No existe de momento un estándar oficial definido que se tenga que cumplir para la publicación de los Open Data. Esto implica un esfuerzo bastante más grande por parte de todas aquellas instituciones que deseen crear servicio en base a estos datos. Un ejemplo bastante claro, es la aplicación para teléfonos móviles "Madrid Bus", por una parte la EMT (Empresa Municipal de Transporte) representa al Gobierno y la Sociedad está representada por aquellas personas que tienen en su teléfono móvil la aplicación "Madrid Bus" instalada. Esta aplicación, habitualmente la usamos para conocer el tiempo estimado de llegada de un bus, entre otras cosas. Toda esa información, la podemos obtener de su sitio web [38] . En este caso la EMT pone a disposición una plataforma mediante la cual se pueda acceder de manera más sencilla a sus datos, en el caso que nuestra intención sea conocer el tiempo estimado de llegada de un bus a una parada en concreta, se puede obtener en [38]. La Figura 5 muestra la plataforma en la que se encuentra el API del servicio "Madrid Bus".

Sin embargo, el uso de esta plataforma, no está dirigido al ciudadano común. Esta plataforma está dirigida a desarrolladores con el fin de que creen servicios en forma de aplicaciones y sean estas aplicaciones las que den facilidad de uso al ciudadano común.

2.1 Datasets Ambientales en Tiempo Real en España

Actualmente hay varias comunidades en territorio Español que han publicado datasets medio-ambientales [39]. Sin embargo, el objetivo de este trabajo es el desarrollo de un servicio de consulta de datos ambientales en tiempo real**.** Considerando esto, el número de comunidades o regiones que publican sus datasets de datos ambientales en tiempo real se reduce considerablemente. Como ya se ha comentado antes, aún no existen estándares para la publicación de los datasets. Por tanto los datasets se publican en formato totalmente distinto y publican datos concretos sobre localidades o gobiernos concretos que al estar en distintos formatos no permiten la realización de comparativas directas.

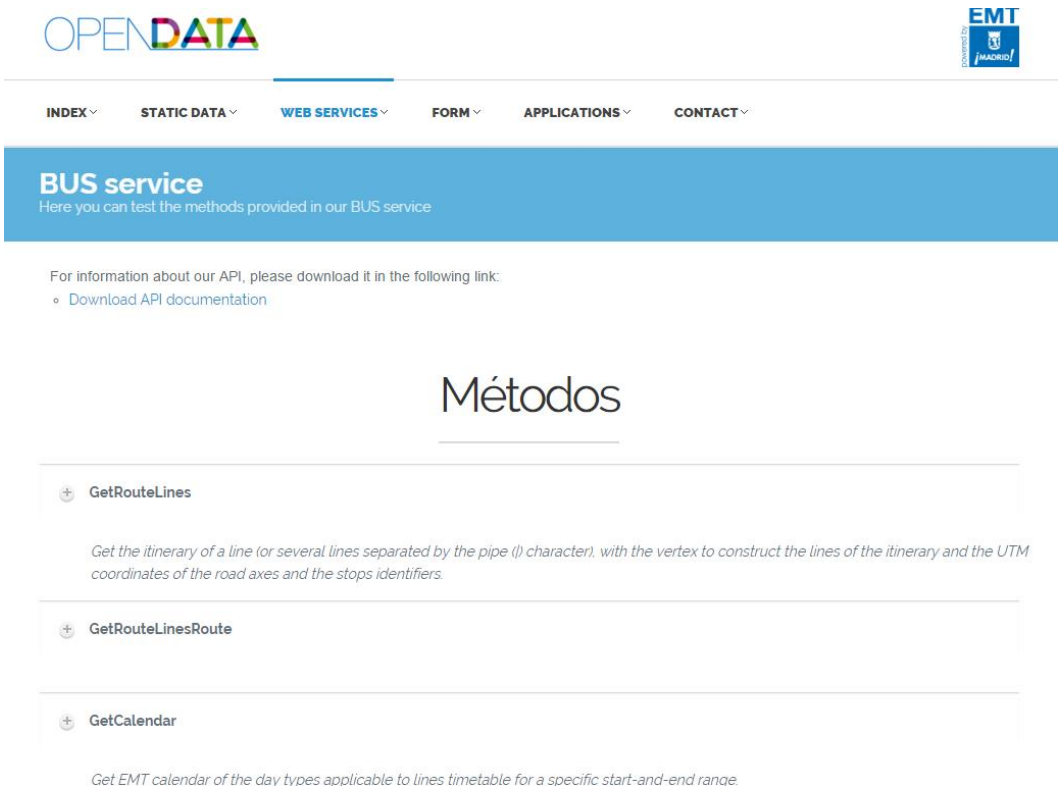


Figura 5. API para el Servicio de Bus Madrid

2.1.1 Datos de la Comunidad de Madrid

Actualmente la Comunidad de Madrid está poniendo a nuestra disposición, un conjunto de datos bajo el título de "Calidad del Aire en Tiempo Real". Estos datos, cumpliendo los requisitos técnicos de Open Data, viene en un formato "machine readable" [40]. En este caso la información viene en un fichero "csv". El fichero está disponible en la web de datos de la Comunidad de Madrid en [41]. El fichero contiene 57 columnas y se actualiza entre los minutos 20 y 30 de cada hora. Este fichero contiene información de diversos indicadores para distintas estaciones. Las estaciones de las que se informa en el fichero pueden verse en la Tabla 1.

Tabla 1. Estaciones dentro de la Comunidad de Madrid

Comunidad Autónoma	Estación	Nombre
28	4	Pza. de España
28	8	Escuelas Aguirre
28	11	Av. Ramón y Cajal

28	16	Arturo Soria
28	17	Villaverde Alto
28	18	Farolillo
28	24	Casa de Campo
28	27	Barajas
28	35	Pza. del Carmen
28	36	Moratalaz
28	38	Cuatro Caminos
28	39	Barrio del Pilar
28	40	Vallecas
28	47	Méndez Álvaro
28	48	P. Castellana
28	49	Retiro
28	50	Pza. Castilla
28	54	Ensanche Vallecas
28	55	Urb. Embajada
28	56	Pza. Fdez. Ladreda
28	57	Sanchinarro
28	58	El Pardo
28	59	Parque Juan Carlos I
28	60	Tres Olivos
28	99	Media de la red

Los indicadores de los que se informa en el fichero se muestran en la Tabla 2.

Tabla 2. Indicadores medidos dentro de la Comunidad de Madrid

Comunidad Autónoma	**Indicador**	**Nombre**
28	1	Dióxido de Azufre
28	6	Monóxido de Carbono
28	7	Monóxido de Nitrógeno
28	8	Dióxido de Nitrógeno
28	9	Partículas < 2.5 um

28	10	Partículas < 10 um
28	12	Óxidos de Nitrógeno
28	14	Ozono
28	20	Tolueno
28	30	Benceno
28	35	Etilbenceno
28	37	Metaxileno
28	38	Paraxileno
28	39	Ortoxileno
28	42	Hidrocarburos Totales (hexano)
28	43	Hidrocarburos (Metano)
28	44	Hidrocarburos No Metánicos (hexano)
28	80	Radiación Ultravioleta
28	81	Velocidad de Viento
28	82	Dirección de Viento
28	83	Temperatura
28	86	Humedad Relativa
28	87	Presión
28	88	Radiación Solar
28	89	Precipitación
28	92	Lluvia ácida

La Tabla 3 muestra la estructura del fichero que se puede descargar.

Estos datos se publican en tiempo real, la actualización de los datos se realiza cada hora. En el fichero sólo vendrán dados los valores de medición menores o iguales a la hora actual X, las columnas correspondientes a horas mayores que X se informarán con valores de medición a 0 y la verificación a “N”. La url del fichero siempre será la misma [41]. La Figura 6 muestra un ejemplo del fichero de datos ambientales en tiempo real de la Comunidad de Madrid a fecha-hora: 25-09-2015 06:30:00.

Tabla 3. Estructura del fichero descargado para la Comunidad de Madrid

Col.	Descripción	Longitud
1	Código de Comunidad Autónoma (Siempre “28” : Comunidad de Madrid)	2
2	Código de Ciudad (Siempre “079”: Madrid)	3
3	Código de Estación (Ver Tabla 1)	3
4	Código de Indicador (Ver Tabla 2)	2
5	Código de Técnica Analítica	2
6	Código de Periodo de Análisis (Siempre “02”: Diario)	2
7	Año	4
8	Mes	2
9	Día	2
10	Valor de la Medición a la hora 01:00	5
11	Verificación del Valor de la Medición a la hora 01:00 (V/N)	1
12	Valor de la Medición a la hora 02:00	5
13	Verificación del Valor de la Medición a la hora 02:00 (V/N)	1
14	Valor de la Medición a la hora 03:00	5
15	Verificación del Valor de la Medición a la hora 03:00 (V/N)	1
…	Valor de la Medición a la hora X	5
…	Verificación del Valor de la Medición a la hora X (V/N)	1
54	Valor de la Medición a la hora 23:00	5
55	Verificación del Valor de la Medición a la hora 23:00 (V/N)	1
56	Valor de la Medición a la hora 24:00	5
57	Verificación del Valor de la Medición a la hora 24:00 (V/N)	1

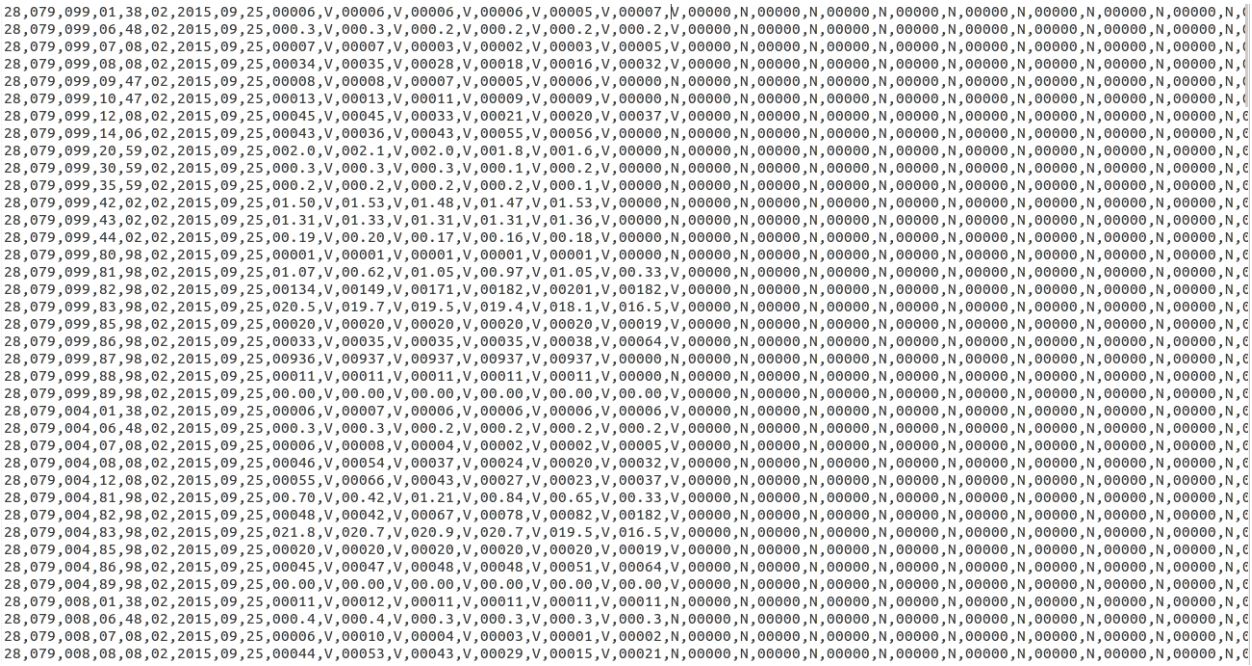


Figura 6. Apariencia del fichero físico descargado para la Comunidad de Madrid

2.1.2 Datos de la Junta de Andalucía

La Junta de Andalucía, fue una de las primeras instituciones en España en hacer públicos en tiempo real los datos ambientales de su región. Dentro de su portal [42], los datos los podemos ver en formato de tabla, pero también permite obtener los datos en formato "machine-readable".

El portal muestra datos de la Red Hidrosur. Mediante ese portal, podemos consultar datos como la temperatura o la pluviometría rellenando un simple formulario en el que nos piden fecha inicial, fecha final y la estación. Esta consulta nos devolverá una tabla pero también tendremos la posibilidad de pasarlo a un fichero plano. La Red Hidrosur consta de un centenar de estaciones de medición dotadas con el equipamiento necesario para la adquisición de los datos y su transmisión en tiempo real al centro de control, donde se reciben y analizan. Se trata de medir indicadores hidrometeorológicos -niveles de agua, caudales, pluviometría, nivometría-. Todos los datos procedentes de las estaciones se transmiten al centro de control ubicado en el edificio de la Cuenca Mediterránea Andaluza, en Málaga. Las estaciones consideradas para las mediciones se muestran en la Tabla 4.

Tabla 4. Estaciones dentro de la Junta de Andalucía

Comunidad Autónoma	Estación	Nombre
2	1	Sierra Mijas (MA)

2	2	Sierra de Luna (CA)
2	3	Embalse de Charco Redondo (CA)
2	6	Los Reales (MA)
2	7	Deposito DI-1 (CA)
2	16	Embalse de la Concepción (MA)
2	22	Málaga - Palacio de la Tinta (MA)
2	27	Ronda (MA)
2	28	Laguna de Fuente Piedra (MA)
2	29	Embalse del Guadalteba (MA)
2	34	Azud de Paredones (MA)
2	37	Embalse de la Vinuela (MA)
2	44	Torrox (MA)
2	47	Lujar (GR)
2	57	Bayarcal (AL)
2	59	Murtas (GR)
2	60	Motril (GR)
2	62	Cerro Canuelo (GR)
2	65	Lanjarón (GR)
2	66	Capileira (GR)
2	67	Río Trevelez (GR)
2	68	Puerto de la Ragua (AL)
2	75	Sierra Alhamilla (AL)
2	76	Sierra de Gador (AL)
2	80	Sierra de los Filabres (AL)
2	81	Oria (AL)
2	82	Tahal (AL)
2	88	El Saltador (AL)
2	89	Almería (AL)
2	91	Ohanes (AL)
2	92	Finana (AL)
2	97	Nijar (AL)
2	103	Río Guadiaro (Majaceite) (MA)
2	104	Río Grande (Las Millanas) (MA)

2	130	Río Guadalhorce (Archidona) (MA)

Los indicadores de los que se informa en el fichero se muestran en la Tabla 5. Todos estos datos pertenecen al Sistema Automático de Información Hidrológica (SAIH) de la Cuenca Mediterránea Andaluza. Este sistema pertenece a Hidrosur cuya web se muestra en la Figura 7. En esta figura, al hacer clic sobre el enlace "Arch. Txt" se generará un fichero de texto con los datos consultados. En dicho fichero se usará el ";" (punto y coma) como separador de campos. Este fichero contiene las columnas según se muestran en la Tabla 6.

Tabla 5. Indicadores medidos para la Junta de Andalucía

Comunidad Autónoma	Indicador	Nombre
2	83	Temperatura
2	84	Pluviometría

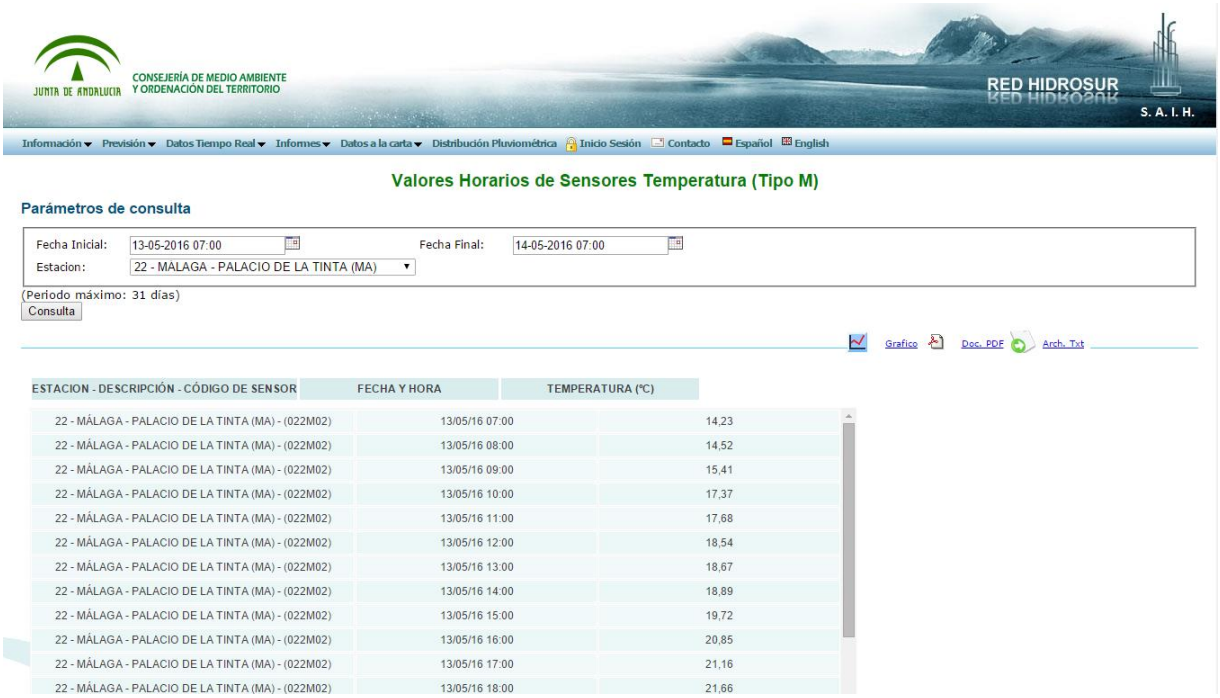


Figura 7. Servicio Web para descargar datos de la Red Hidrosur

Tabla 6. Estructura del fichero descargado para la Junta de Andalucía.

Col.	Descripción	Longitud
1	Código de Estación – Descripción de Estación – Código de sensor (Ver Tabla 4)	Delimitado por separador

2	Fecha y Hora	Delimitado por separador
3	Valor de Medición	Delimitado por separador

El primer campo contiene información del "Código de Sensor" que para el alcance de este proyecto, no es necesario. Por tanto, sólo mostraremos en esta tabla el código y descripción de cada estación para esta región. Estos datos se publican en tiempo real. Los datos se actualizan cada hora.

Un detalle que habrá que solucionar es la automatización de la descarga debido a que, primero es necesario completar los campos *fecha inicio, fecha fin y estación,* y luego, al pulsar sobre "Arch. Txt", se genera finalmente el fichero plano para descargarlo. Esto sería muy complejo de automatizar (aunque es posible automatizarlo, con herramientas Open-Source sería bastante complejo).

Una alternativa bastante sencilla, aprovechando que la consulta al servidor de datos se realiza con el método "GET", es indicar los parámetros en la URL. Esto se explicará a detalle en la sección correspondiente a la implementación del proceso de integración para este dataset (Capítulo 3).

La Figura 8 muestra un ejemplo del fichero de datos ambientales en tiempo real de la Junta de Andalucía entre el 07-Mar-2016 y el 10-Mar-2016 para la estación: Embalse Charco Redondo. Visualmente pueden observarse grandes diferencias respecto a la Figura 6 de los datos de la CAM.

```
Estación - Descripción - Código de sensor; Fecha Hora; Temperatura ºC;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 00:00; 10,590;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 01:00; 10,200;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 02:00; 10,070;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 03:00; 9,560;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 04:00; 9,110;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 05:00; 8,630;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 06:00; 8,380;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 07:00; 8,220;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 08:00; 8,590;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 09:00; 9,090;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 10:00; 10,120;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 11:00; 11,250;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 12:00; 12,770;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 13:00; 12,400;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 14:00; 10,860;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 15:00; 11,090;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 16:00; 13,980;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 17:00; 15,610;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 18:00; 16,340;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 19:00; 14,760;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 20:00; 13,350;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 21:00; 12,490;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 22:00; 11,840;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 07/03/16 23:00; 11,190;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 00:00; 10,710;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 01:00; 10,550;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 02:00; 9,930;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 03:00; 9,490;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 04:00; 9,140;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 05:00; 8,740;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 06:00; 8,480;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 07:00; 8,490;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 08:00; 8,080;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 09:00; 7,870;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 10:00; 9,920;
3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02); 08/03/16 11:00; 12,110;
```

Figura 8. Ejemplo de fichero descargado de la Red Hidrosur

2.2 Formato de Integración

El objetivo de esta sección es mostrar una estructura de datos que será el "output" del proceso de integración de ambos datasets: Comunidad de Madrid y Junta de Andalucía. Antes de empezar a desarrollar el proceso de integración, es fundamental saber el formato de la estructura de datos, para esto tenemos que analizar dos puntos:

Alcance del Trabajo. El proceso de integración puede llegar a ser muy complejo considerando la cantidad de información que existe en cada dataset y los distintos formatos en que podemos encontrar los datos. Es importante saber que datos son los que realmente se quieren integrar y que datos no se quieren integrar para disminuir la amplitud de nuestros datos.

Información Común. Los datasets que queremos integrar deben tener información en común sin importar el formato. La integración significa combinar los mismos indicadores con distintos formatos en un único archivo. Es recomendable analizar los datasets para encontrar solo la información que realmente nos interesa integrar.

Técnicamente, lo que se busca es crear una estructura de datos sobre la cual se vayan añadiendo datos sin importar si vienen del dataset de la Comunidad de Madrid o de la Junta de Andalucía. En este caso, la estructura final de integración es un fichero plano (csv). Para conseguir esto, previamente estos datos han pasado un proceso de "formateo".

La estructura final del fichero de datos de integración tiene las columnas como se muestra en la Tabla 7.

Tabla 7. Estructura final del formato de integración.

Col.	Descripción	Longitud
1	Comunidad Autónoma (28 : Comunidad de Madrid, 2 : Junta de Andalucía)	Delimitado por separador
2	Ciudad (79 : Comunidad de Madrid, 2 : Junta de Andalucía)	Delimitado por separador
3	Estación (Unión de Tabla 1 y Tabla 4)	Delimitado por separador
4	Indicador (Unión de Tabla 2 y Tabla 5)	Delimitado por separador
5	Año	Delimitado por separador
6	Mes	Delimitado por separador

7	Día	Delimitado por separador
8	Hora	Delimitado por separador
9	Valor de Medición	Delimitado por separador

La apariencia del fichero final después del proceso de integración es la mostrada en la Figura 9. Se trata de un formato sencillo que permite su descarga y posterior automatización con gran rapidez.

```
comaut,ciudad,estaci,parame,ano,mes,dia,horavl,valorl
28,79,4,83,2015,9,25,0,21.8
28,79,4,83,2015,9,25,1,20.7
28,79,4,83,2015,9,25,2,20.9
28,79,4,83,2015,9,25,3,20.7
28,79,4,83,2015,9,25,4,19.5
28,79,4,83,2015,9,25,5,16.5
28,79,4,83,2015,9,25,6,13.7
28,79,4,83,2015,9,25,7,14.8
28,79,4,83,2015,9,25,8,16
28,79,4,83,2015,9,25,9,16.3
28,79,4,83,2015,9,25,10,21
28,79,4,83,2015,9,25,11,23.6
28,79,4,83,2015,9,25,12,26.7
28,79,4,83,2015,9,25,13,28.2
28,79,4,83,2015,9,25,14,28.8
28,79,4,83,2015,9,25,15,29.5
28,79,4,83,2015,9,25,16,29.5
28,79,4,83,2015,9,25,17,29.5
28,79,4,83,2015,9,25,18,28.3
28,79,18,83,2015,9,25,0,20.9
28,79,18,83,2015,9,25,1,19.8
28,79,18,83,2015,9,25,2,19.4
28,79,18,83,2015,9,25,3,19.6
28,79,18,83,2015,9,25,4,17.7
28,79,18,83,2015,9,25,5,16.9
28,79,18,83,2015,9,25,6,16.5
28,79,18,83,2015,9,25,7,16.3
28,79,18,83,2015,9,25,8,16.2
28,79,18,83,2015,9,25,9,18.1
28,79,18,83,2015,9,25,10,21.4
28,79,18,83,2015,9,25,11,24.1
28,79,18,83,2015,9,25,12,26
28,79,18,83,2015,9,25,13,27.1
28,79,18,83,2015,9,25,14,27.1
28,79,18,83,2015,9,25,15,26.8
28,79,18,83,2015,9,25,16,26.8
28,79,18,83,2015,9,25,17,26.7
```

Figura 9. Ejemplo de fichero final posterior al proceso de integración

Capítulo 3. Proceso de Integración

En este capítulo se describe el proceso de integración que es un proceso automático, que se ejecuta en tiempo real y cuyo objetivo es transformar los datos de los distintos datasets con formatos variables en un único formato (Tabla 7). Esta tarea se realiza utilizando procesos de integración ad hoc para cada uno de los datasets que se desee incorporar, por tanto el proceso de integración para un dataset en concreto realiza tareas muy distintas al proceso de integración de otro dataset. Técnicamente, el proceso de integración se compone de 4 subprocesos los cuales se muestran en la Tabla 8.

Tabla 8. Sub-procesos del proceso de integración

Número	**Sub-Proceso**	**Descripción**
1	Descarga	Se descarga el dataset publicado desde la Web Institucional
2	Carga a Estructura Temporal	Se carga los datos del dataset en una estructura temporal para posteriormente procesarlo
3	Verticalización o Formateo	Realiza las comprobaciones, validaciones y/o formateos necesarios antes de proceder a almacenar los nuevos datos
4	Inserción en Estructura Final	Finalmente se almacenan los datos en el formato especificado en la Tabla 7.

Tabla 9. Versiones de la implementación del proceso de integración

dataset	**Versión del proceso de integración**	**Subprocesos**			
		Descarga	**Carga de Estructura Temporal**	**Verticalización y/o Formateo**	**Inserción en Estructura Final**
Comunidad de Madrid	Versión Java – Oracle	Java	Oracle	Oracle (Verticalización y Formateo)	Oracle
	Versión R	R	R	R (Verticalización y Formateo)	R
Junta de Andalucía	Versión R	R	R	R (Formateo)	R

Se han utilizado 3 tecnologías para implementar estos subprocesos: Java, Oracle y R. La Tabla 9 muestra la tecnología con la que ha implementado cada subproceso así como las versiones realizadas del proceso de integración. En las siguientes secciones de este capítulo se exponen los detalles de la integración aplicada a datos procedentes de los Open Data de la Comunidad de Madrid y de la Junta de Andalucía.

3.1 Integración del dataset de la Comunidad de Madrid

El proceso de integración del dataset de la Comunidad de Madrid es el que implica mayor análisis. Como se observa en la Tabla 9, se han realizado dos versiones de este proceso, una es usando tecnologías Java y Oracle mientras que la otra usa totalmente tecnologías Open-Source con R.

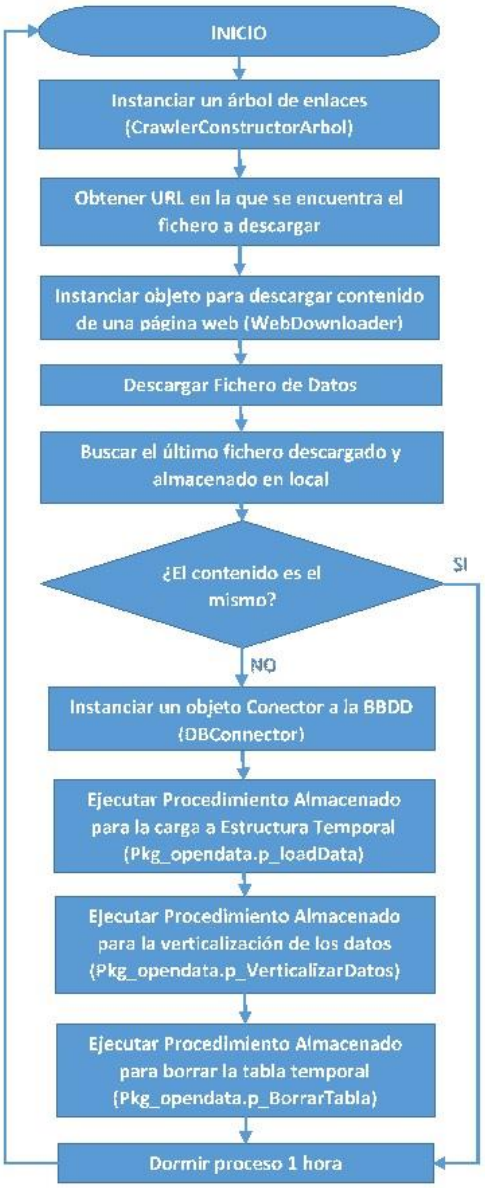


Figura 10. Diagrama de flujo de la versión Java-Oracle

3.1.1 Versión Java-Oracle.

En esta sección se detalla a nivel técnico lo que se realiza mediante tecnología Java y lo que se ha desarrollado con tecnología Oracle. Para esta versión, el programa base está desarrollado en Java, y desde este programa, se van realizando las llamadas a cada uno de los subprocesos que están implementados con tecnología Oracle. La Figura 10 muestra el diagrama de flujo para la implementación de esta versión.

Descarga – Java

Un punto importante a mencionar y que se ha tenido que contemplar es el posible cambio de URL de descarga del fichero del dataset. Para solucionar este posible evento, se ha tenido que usar herramientas que permitan hacer "Web Crawling" [43]. Un proceso Web Crawling se usa para indexar las páginas web dentro de un dominio para luego hacer una búsqueda en todos estos elementos indexados.

En este trabajo de fin de Máster se ha implementado un proceso en Java que realiza "Web Crawling" para que busque la URL donde se encuentra el dataset sin importar si su ubicación es variable. Este proceso elabora un árbol de enlaces (URL's) desde una URL base, para luego en cada uno de los elementos del árbol buscar un patrón.

Para este sub-proceso he implementado 3 clases java: Clase *CrawlerNodo*, *CrawlerConstructorArbol* y *WebDownloader***.** La Tabla 10, la Tabla 11 y la Tabla 12 detallan las funciones de estas clases respectivamente.

1. **Clase CrawlerNodo:** La clase *CrawlerNodo* representa un nodo en el árbol de enlaces.

Tabla 10. Funciones de la clase CrawlerNodo

Nº	**Método**	**Descripción**
1	**Constructor**	Crea un nodo del árbol.

2. **Clase CrawlerConstructorArbol:** Mediante esta clase se crea el árbol de enlaces y se encuentra aquel enlace que contenga un patrón en concreto. Para crear el árbol, se añade como primer nodo la URL base. Luego, se busca el contenido de esta URL base, si esta tiene enlaces, estos enlaces se añaden al árbol. Es decir se van añadiendo más nodos al árbol a medida que se va recorriendo cada nodo.

Tabla 11. Funciones de la clase CrawlerConstructorArbol

Nº	Método	Descripción
1	**Constructor**	Esta clase tiene un constructor que se encarga de crear un primer nodo con la URL base que se envía como parámetro.
2	**ConstruirArbol**	Este método se encarga de generar el árbol de enlaces de un tamaño establecido por la profundidad enviada por parámetro. La profundidad indica el nivel de descendencia respecto al primer nodo (URL base).
3	**get_ProfundidadIndice**	Este método devuelve el nivel de profundidad de un nodo dentro del árbol de enlaces.
4	**fill_tree**	Este método es usado para añadir más nodos (enlaces) al árbol de enlaces en el índice indicado como parámetro.
5	**get_urlContainedWord**	Este método devuelve el nodo (enlace) el cual contenga una cadena de caracteres enviada por parámetro (un patrón de búsqueda).
6	**print_tree** **print_childs**	Métodos usados para imprimir el árbol.

La Figura 11 muestra el diagrama de clases de CrawlerConstructorArbol y CrawlerNodo, mientras que la Figura 12 muestra un esquema del árbol generado con estas clases.

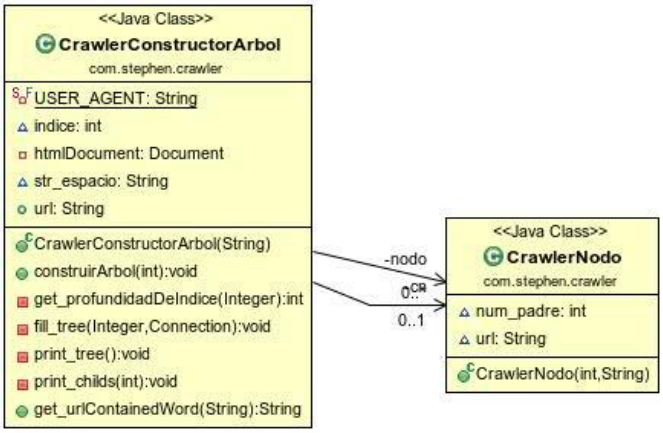
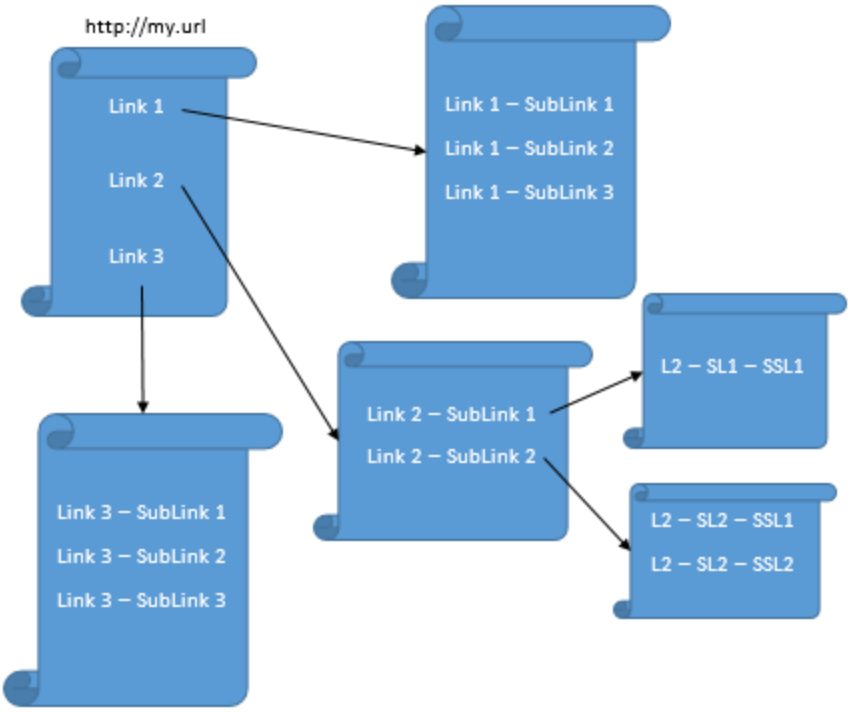


Figura 11. Diagrama de clases CrawlerConstructorArbol y CrawlerNodo



El árbol generado:

Nodo	Profundidad	Link
1	1	http://my.url
2	2	Link 1
3	3	Link 1 – SubLink 1
4	3	Link 1 – SubLink 2
5	3	Link 1 – SubLink 3
6	2	Link 2
7	3	Link 2 – SubLink 1
8	4	L2 – SL1 – SSL1
9	3	Link 2 – SubLink 2
10	4	L2 – SL2 – SSL1
11	4	L2 – SL2 – SSL2
12	2	Link 3
13	3	Link 3 – SubLink 1
14	3	Link 3 – SubLink 2
15	3	Link 3 – SubLink 3

Figura 12. Ejemplo de árbol generado por un objeto CrawlerConstructorArbol

3. **Clase WebDownloader:** Realiza la descarga y compara ficheros. La Figura 13 muestra el diagrama de clase.

Tabla 12. Funciones de la clase WebDownloader

Nº	Método	Descripción
1	**Constructor**	Esta clase tiene un constructor que instancia un objeto URL con la dirección que le enviamos por parámetro y almacena el nombre de fichero enviado como parámetro que será usado para nombrar al fichero descargado.

2	**getFilePath**	Devuelve la ruta del fichero descargado
3	**DownloadWeb**	Descargamos el contenido de la URL que se pasó como parámetro en el constructor y lo almacenamos en un fichero con nombre enviado también al constructor.
4	**renameandMoveFile**	Método que se encarga de mover un fichero de una ubicación a otra con otro nombre
5	**lastFileModified**	Devuelve el último fichero modificado dentro de un directorio
6	**compareFile**	Recibe dos rutas de ficheros como parámetros y compara dichos ficheros, devuelve 1 si son iguales y 0 si son distintos.

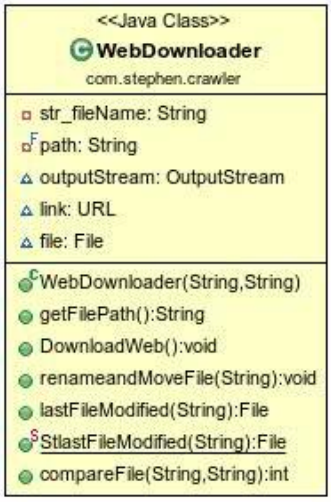


Figura 13. Diagrama de clase WebDownloader

La implementación en detalle de este sub-proceso se adjunta en el Anexo I.

Carga de Estructura Temporal – Oracle

Una vez que el fichero está descargado ya podemos aplicar cualquier tecnología para procesar los datos. El subproceso "Carga de Estructura Temporal" se encarga de pasar los datos descargados a una estructura de datos la cual pueda ser manejada con mayor facilidad por la tecnología con la que se implementará este subproceso.

La tecnología que se usará para implementar este subproceso es Oracle, por tanto, es necesario tener configurada una BBDD Oracle. Se asumirá que se tiene ya configurada la BBDD con nombre *open data*. Dentro de esta BBDD se tendrá que crear una tabla temporal

para almacenar los datos que inicialmente se encuentran en el fichero.

La tabla temporal, que para efecto de la implementación se llamará **T_TEMPDATA,** tendrá exactamente la misma estructura que el fichero como se muestra en la Tabla 13.

Tabla 13. Estructura de la tabla temporal T_TEMPDATA

Nº	**Campo**	**Tipo**
1	FIELD01	VARCHAR2(5)
2	FIELD02	VARCHAR2(5)
3	FIELD03	VARCHAR2(5)
...	FIELDXX	VARCHAR2(5)
55	FIELD55	VARCHAR2(5)
56	FIELD56	VARCHAR2(5)
57	FIELD57	VARCHAR2(5)

Sobre esta tabla se vuelcan los datos que tenemos en el fichero. Este proceso se puede realizar con herramientas de Oracle como el "SqlLoader". Sin embargo, para tenerlo automatizado, este subproceso será implementado con procedimientos almacenados de Oracle. Para esta implementación el programa base sigue siendo Java ya que estamos en la versión Java-Oracle. Por tanto, la implementación de este subproceso será ejecutada desde Java. Sin embargo, en esta sección detallaremos procesos Oracle necesarios que nos sirven para almacenar y extraer información desde una base de datos. La implementación se detalla en la siguiente sección.

1. **Paquete Pkg_OpenData:** Para empezar a crear los procedimientos almacenados, tenemos que crear un paquete Oracle, para efectos de la implementación, el paquete se llamará *Pkg_OpenData*. El lenguaje de Oracle para crear procedimientos almacenados es Procedural Language – Structured Query Language (PL/SQL) y los procedimientos de este paquete son los mostrados en la Tabla 14.

Tabla 14. Procedimiento de Pkg_OpenData para la "Carga de Estructura Temporal"

Nº	Procedimientos	Descripción
1	**p_LoadData**	Recibirá por parámetro el nombre del fichero a cargar, y cada línea del fichero lo insertará en una tabla T_TEMPDATA
2	**p_SplitLine**	Procedimiento recursivo que recibe como parámetros una línea del fichero y un carácter que sirva como separador y los convierte en valores para que se construya la sentencia INSERT INTO.

La implementación de este subproceso se realiza en código Java ya que estamos en la versión Java-Oracle, en el cual el programa base está hecho en Java. Dentro de esta implementación se hace uso de una clase creada para la conexión a BBDD Oracle y la ejecución de funciones y procedimientos dentro de ella. Esta clase se llama *DBConnector.*

2. **Clase DBConnector:** Esta clase se encarga de establecer una conexión a una base de datos indicada como parámetro y contiene funciones para ejecutar procedimientos almacenados en la base de datos. La Tabla 15 muestra el detalle de las funciones de esta clase y la Figura 14 muestra el diagrama de clase.

Tabla 15. Funciones de la clase DBConnector

Nº	Método	Descripción
1	**Constructor**	Esta clase tiene un constructor que asigna los parámetros recibidos a variables usadas para hacer la conexión mediante la función ***connect***.
2	**Connect**	Realiza la conexión a la base de datos
3	**Exec_StoreProcedureOneStrPar**	Ejecuta un procedimiento almacenado que recibe solo un único parámetro de tipo String
4	**Exec_StoreProcedureNoPars**	Ejecuta un procedimiento almacenado que no recibe parámetros.
5	**Disconnect**	Termina la conexión con la base de datos.

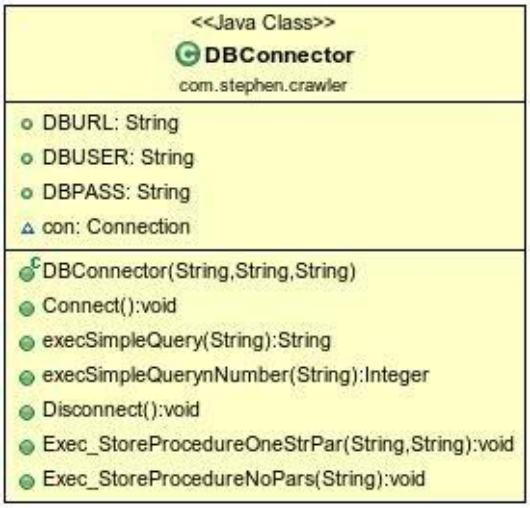


Figura 14. Diagrama de clase DBConnector

El detalle de la implementación de este sub-proceso se adjunta en el Anexo II.

Verticalización y Formateo - Oracle

En esta sección se detallará una manera de cómo convertir los datos de la tabla T_TEMPDATA que ha sido cargada en el subproceso previo, en una estructura de datos definida en la Tabla 7. Para esta implementación el programa base sigue siendo Java ya que estamos en la versión Java-Oracle. Por tanto, la implementación de este subproceso será ejecutada desde Java. Sin embargo, en esta sección detallaremos procesos Oracle necesarios que nos sirven para almacenar y extraer información desde una Base de Datos. La implementación se detalla en la siguiente sección.

Para conseguir la estructura de la Tabla 7 (Estructura Final) a partir de la tabla T_TEMPDATA es necesario aplicar un proceso que se llamará "Verticalización". La función de este proceso es a partir de cada registro dentro de la tabla T_TEMPDATA, generar X registros que se insertarán en la estructura final, en donde X es menor o igual que 24.

Antes de la verticalización, los datos tienen la forma que muestra la Figura 15. Después de la verticalización, los datos tendrán la forma mostrada en la Figura 16. Este proceso se ha desarrollado con un procedimiento almacenado de Oracle y para efectos de implementación se llamará p_VerticalizacionDatos. En la

Tabla 16 se detalla este procedimiento almacenado que está contenido dentro del paquete Pkg_OpenData. El detalle de la implementación de este sub-proceso se adjunta en el Anexo III.

Columna	1	2	3	4	5	6	7	8	9	10	11	12
Campo	Comunidad Autónoma	Ciudad	Estación	Indicador	Técnica Analítica	Periodo Analisis	Año	Mes	Día	Hora 01	V/N	Hora 02
Valor	28	079	099	01	38	02	2015	09	25	00006	V	00006

Columna	13	14	15	16	17	18	19	20	21	22	23	24
Campo	V/N	Hora 03	V/N	Hora 04	V/N	Hora 05	V/N	Hora 06	V/N	Hora 07	V/N	Hora 08
Valor	V	00006	V	00006	V	00005	V	00007	V	0	N	0

Columna	25	26	27	28	29	30	31	32	33	34	35	36
Campo	V/N	Hora 09	V/N	Hora 10	V/N	Hora 11	V/N	Hora 12	V/N	Hora 13	V/N	Hora 14
Valor	N	0	N	0	N	0	N	0	N	0	N	0

Columna	37	38	39	40	41	42	43	44	45	46	47	48
Campo	V/N	Hora 15	V/N	Hora 16	V/N	Hora 17	V/N	Hora 18	V/N	Hora 19	V/N	Hora 20
Valor	N	0	N	0	N	0	N	0	N	0	N	0

Columna	49	50	51	52	53	54	55	56	57
Campo	V/N	Hora 21	V/N	Hora 22	V/N	Hora 23	V/N	Hora 24	V/N
Valor	N	0	N	0	N	0	N	0	N

Figura 15. Registro de ejemplo dentro del fichero de la Comunidad de Madrid

Comunidad Autónoma	Ciudad	Estación	Parámetro	Año	Mes	Día	Hora	Valor
28	079	099	01	2015	09	21	1	6
28	079	099	01	2015	09	21	2	6
28	079	099	01	2015	09	21	3	6
28	079	099	01	2015	09	21	4	6
28	079	099	01	2015	09	21	5	5
28	079	099	01	2015	09	21	6	7

Figura 16. Registro verticalizado

Tabla 16. Procedimiento de Pkg_OpenData para la "Verticalización y Formateo"

Nº	Procedimientos	Descripción
3	**p_ VerticalizacionDatos**	Este procedimiento no recibe ningún parámetro de entrada ya que la única fuente de entrada es la tabla T_TEMPDATA que ya está cargada y cuyos datos serán "verticalizados" por este procedimiento. Inserta para cada hora un registro en la estructura final con los datos que serán comunes para todas las horas. Los datos comunes son: Comunidad Autónoma, Ciudad, Estación, Parámetro, Técnica Analítica, Período de Análisis, Año, Mes y Día (columna 0 a 9 en la tabla T_TEMPDATA). Ejecuta internamente el procedimiento ***p_EjecutarInsertControlado***.

Inserción en Estructura Final - Oracle

En esta sección se detalla el procedimiento *p_EjecutarInsertControlado* que se ha llamado desde el procedimiento que realiza la verticalización. Este procedimiento ejecuta las sentencias insert necesarias para cargar la estructura final de datos. Además, se volverá al programa base implementado en Java, para cerrar la conexión a la base de datos.

Un detalle importante a mencionar es que los valores de medición que han sido cargados en una hora X, pueden modificarse en algún fichero que se descargue en una hora X+Y. Para llevar un control de qué cambios se han realizado en datos anteriores, este subproceso ofrece un control de errores informado en un fichero de log.

Para esta versión, la estructura final de datos será una tabla con la estructura mostrada en la Tabla 17. Para efectos de implementación esta tabla se llamará **T_DATOS**:

Tabla 17. Tabla Oracle sobre la que se volcará la estructura final

Nº	**Campo**	**Tipo**
1	PK	NUMBER (NO NULL)
2	FK_COMUNIDADAUTONOMA	NUMBER(4)
3	FK_CIUDAD	NUMBER(4)
4	FK_ESTACION	NUMBER(4)
5	FK_PARAMETRO	NUMBER(4)
6	FK_TECNICAANALITICA	NUMBER(4)
7	PERIODOANALISIS	NUMBER(2)
8	ANHO	NUMBER(4)
9	MES	NUMBER(2)
10	DIA	NUMBER(2)
11	HORA	NUMBER(2)
12	MINUTO	NUMBER(2)
13	SEGUNDO	NUMBER(2)
14	VALOR	NUMBER(20,5)
15	VALIDACION	VARCHAR2(1)

Esta tabla deberá tener una *constraint* de tipo “UNIQUE” por los campos siguientes:

- FK_COMUNIDADAUTONOMA
- FK_CIUDAD
- FK_ESTACION
- FK_PARAMETRO
- FK_TECNICANALITICA
- ANHO
- MES
- DIA
- HORA

Lo que se consigue con esta *constraint* es que el conjunto de todos estos valores deberá ser único para cada registro al momento de insertar o actualizar la tabla T_DATOS, lanzando un mensaje de error en caso de que se intente insertar (o actualizar) algún registro que incumpla esta restricción. Para el caso de esta implementación, esto restringe que haya registros repetidos en la tabla T_DATOS. Se puede observar que son los campos comunes para todos los registros de una misma hora.

Para incluir esta restricción a la tabla T_DATOS, se debe incluir la siguiente constraint de nombre **“**CST_UNIQUE_DATOS” en su creación:

```
CONSTRAINT "CST_UNIQUE_DATOS"
UNIQUE ("ANHO", "MES", "DIA", "FK_COMUNIDADAUTONOMA", "FK_CIUDAD", "FK_ESTACION", "HORA",
"FK_PARAMETRO", "FK_TECNICAANALITICA")
```

Lo más importante de este subproceso se realiza en el procedimiento *p_EjecutarInsertControlado* (invocado en la sección “Verticalización y Formateo - Oracle”). Dentro de este procedimiento se ejecutan las funciones: *f_toNumberCust, f_BorrarRegistroDatos* y *p_BorrarTabla* que estarán contenidos en el paquete *Pkg_OpenData.* En la Tabla 18 se detalla cada una de estas funciones. Con la ejecución del procedimiento *p_EjecutarInsertControlado* se insertan finalmente los datos en la estructura final, es decir la tabla T_DATOS, e incluso, tendríamos un control de valores de medición modificados. Los datos estarían ya así integrados**.**

Tabla 18. Procedimiento de Pkg_OpenData para la "Inserción en Estructura Final"

Nº	Procedimientos	Descripción
4	**p_EjecutarInsertControlado**	Este procedimiento intenta realizar el insert en la tabla T_DATOS. Si el insert falla por motivos de la constraint "CST_UNIQUE_DATOS" y se detecta que ha habido una modificación de los valores de medición, se inserta en un Log el hecho.
5	**f_toNumberCust**	Esta función realiza una conversión de la columna de valores de medición que originalmente se recogen como cadena de caracteres y se convierten a números.
6	**f_BorrarRegistroDatos**	Esta función se encarga de borrar los registros cuyo valor de medición ha sido modificado para que se pueda insertar un nuevo registro con el nuevo valor de medición. Devuelve en una cadena de caracteres, los datos que han sido borrados para que se informe posteriormente en log
7	**p_BorrarTabla**	Este procedimiento, no es propio del subproceso de Inserción en la tabla T_DATOS, pero es importante para limpiar la tabla T_TEMPDATA de datos ya procesados. Este procedimiento simplemente borra la tabla T_TEMPDATA

Para finalizar esta versión, queda pendiente una pequeña implementación en el programa base Java. El detalle de la implementación de este sub-proceso se adjunta en el Anexo IV. Para la implementación de esta versión he desarrollado 5 clases las cuales interactúan entre sí para lograr la integración de los datos. En la Figura 17 se muestra el diagrama de clases para todas las clases que he construido.

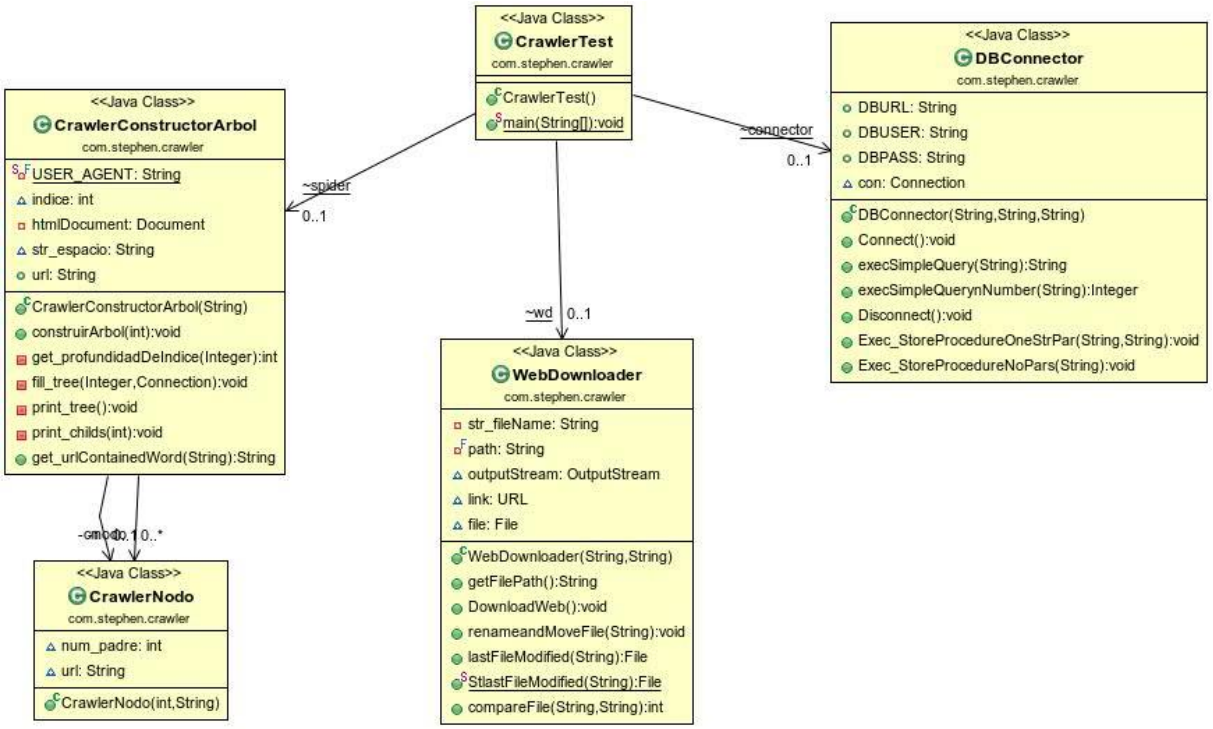


Figura 17. Diagrama de clases. Versión Java-Oracle

3.1.2 Versión R

En esta sección se detalla a nivel técnico el proceso de integración para el dataset de la Comunidad de Madrid mediante tecnología Open-Source, en este caso el lenguaje R. Para esta versión la estructura final será un *data frame* de R que será volcando a un fichero.

Si bien es cierto que esta versión consta de un único script, existen ciertas características que han ido cambiando a lo largo del desarrollo del proyecto con el fin de optimizar el rendimiento. Por tanto de esta versión existen 3 subversiones con características muy específicas para cada una. Estas características se describen en la Tabla 19. Las versiones son las mostradas en la Tabla 20 y se han elaborado conforme a como se ha investigado y en base a algunas sugerencias de personas especialistas en esta tecnología acerca de maneras de optimizar el proceso. La versión *OpenDat_Dist.r* fue la primera versión que se elaboró y *OpenData_Ofh.r* es la última.

A continuación se detalla cada uno de los 4 subprocesos de integración: “Descarga” (excepto en la primera y segunda subversión), “Carga de Estructura Temporal”, “Verticalización y Formateo” e “Inserción en Estructura Final” para cada una de las versiones mostradas en la Tabla 20.

Tabla 19. Aspectos considerados para implementar el proceso de integración

Nº	Característica	Descripción
1	Distribución	Significa que la estructura final de datos puede ser una sola, o pueden ser varias distinguidas por el indicador.
2	Control de Cambios	Significa que se genera un fichero log en el que se informa sobre los cambios en valores de medición anteriores a la hora actual (similar al subproceso "Inserción en Estructura Final – Oracle" en la versión "Java-Oracle" para el dataset de la Comunidad de Madrid).
3	Librerías de Optimización	Significa que se hace uso de librerías R para optimizar el rendimiento del proceso. Estas librerías son: ***reshape2*** y ***dplyr.***
4	Optimización del campo Fecha	En las versiones previas, la fecha se compone de 4 campos: año, mes, día y hora y son estos 4 valores los que se almacenan en la estructura final. Esta característica significa almacenar construir una fecha en base a estos 4 campos y almacenar un sólo campo con formato de fecha en la estructura final.

Tabla 20. Versiones de la implementación en R

Nº	Versión	Característica			
		Distribución	**Control de Cambios**	**Librerías de Optimización**	**Optimización del campo fecha**
1	OpenData_Dist.r	NO	SI	NO	NO
		SI	SI	NO	NO
2	OpenData_Opt.r	SI	NO	NO	NO
3	OpenData_Lib.r	SI	NO	SI	SI

OpenData_Dist.r

Este Script realiza la integración de los datos de forma Distribuida y No Distribuida. La diferencia técnicamente entre uno y otro es que una variable booleana es VERDADERO cuando se desee hacer la integración distribuida y FALSO cuando se desee hacer la integración no distribuida. Para esta versión la estructura final será un fichero csv. La característica principal para esta implementación es que se realiza un control de cambios que se informa en un fichero tipo log.

Para la ejecución distribuida existe una estructura final para cada indicador, es decir existen tantos ficheros de datos como indicadores. Los ficheros usan la siguiente nomenclatura: [Código indicador]_myStoredData.csv, un ejemplo de conjunto de estos ficheros se muestran en la Figura 18.

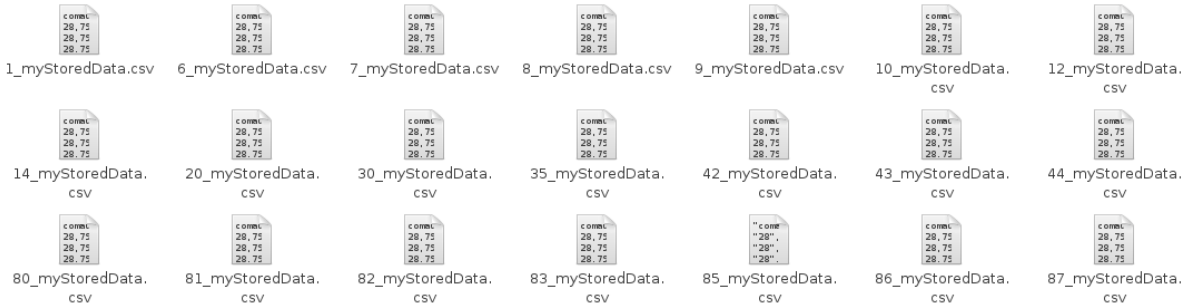


Figura 18. Ficheros generados para una implementación distribuida

Para la ejecución no distribuida existe una única estructura final para todos los indicadores, el nombre de este fichero es: *myStoredData.csv* tal y como se muestra en la Figura 19.

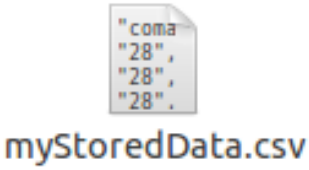


Figura 19. Fichero generado para una implementación no distribuida

Para la implementación de esta versión se asume el hecho de que ya se ha realizado la descarga del fichero. Además para esta implementación es necesario definir previamente dos funciones: *getMaxCurrentDate* y *getMaxStoredDate*. También se define la función: *setNewDataToMSD* que se detalla más adelante. La descripción de estas funciones se detalla

en la Tabla 21

Tabla 21. Funciones iniciales de OpenData_Dist.r

Nº	Procedimientos	Descripción
1	**getMaxCurrentDate**	Esta función obtiene la fecha más alta para todos los indicadores (no distribuido) o para un indicador en concreto (distribuido) dentro del fichero descargado.
2	**getMaxStoredDate**	Esta función obtiene la fecha más alta para todos los indicadores (no distribuido) o para un indicador en concreto (distribuido) dentro de la estructura final ya almacenada.

Esta versión implementa todos los subprocesos indicados en la Tabla 8 (excepto el subproceso de descarga). Las siguientes secciones detallan cada uno de los subprocesos:

Carga de Estructura Temporal – R

Una vez se tenga disponible el fichero descargado, podemos cargarlo a una estructura temporal, en este caso, esta estructura temporal será un *data frame*, que podemos describir como una tabla en memoria.

Tabla 22. Funciones de OpenData_Dist.r para la "Carga de Estructura Temporal"

Nº	Procedimientos	Descripción
3	**setNewDataToMSD**	Esta función evalúa la variable booleana ***bDist*** para realizar una ejecución Distribuida o No Distribuida. Internamente, esta función implementa dos funciones: ***addToMyStoredData*** que realiza lo que es la "Verticalización y Formateo" y ***prepareforDist*** que es una función propia de la ejecución distribuida para gestionar las invocaciones a ***addToMyStoredData***

Verticalización y Formateo - R

Esta sección detalla las instrucciones para convertir el data frame *myCurrentData* en una estructura final (Tabla 7). Para realizar este proceso se ha definido la función *addToMyStoredData* (que se detalla en la Tabla 23) que se encuentra dentro de la función

setNewDataToMSD.

Tabla 23. Funciones de OpenData_Dist.r para la "Verticalización y Formateo"

Nº	Procedimientos	Descripción
4	**addToMyStoredData**	Esta función se ha implementado de manera que sirva para los dos tipos de ejecución: Distribuida y No Distribuida. La finalidad de esta función es construir o escribir sobre un data frame ***my_StoredData*** con la misma apariencia que la estructura final (Tabla 7) para su posterior inserción en la estructura final. Otra finalidad importante de esta función es informar en un fichero Log de los cambios detectados para los datos cargados previamente, es decir se implementa en esta función el **control de cambios**.

Inserción en Estructura Final – R

Para la ejecución distribuida el proceso de inserción en estructura final se realiza dentro de la función *prepareforDist* que se detalla en la Tabla 24. La finalidad de esta función, es realizar las llamadas correspondientes a *addToMyStoredData* para construir un data frame para cada uno de los indicadores que se encuentran en el data frame *myCurrentData*. Después de realizado esto, esta función ejecuta la sentencia *write.csv* que realiza el volcado de *myStoredData* en la estructura final de datos, es decir en el fichero csv.

Tabla 24. Funciones de OpenData_Dist.r para la "Inserción en Estructura Final"

Nº	Procedimientos	Descripción
5	**prepareforDist**	Esta función es propia de la ejecución distribuida y tiene por finalidad realizar las invocaciones a *addToMyStoredData* considerando que el indicador pueda cambiar. Al mismo tiempo realiza la Inserción con sentencias *write.csv* que vuelcan los datos de *myStoredData* al fichero final de datos.

La "Inserción en Estructura Final" también se realiza fuera de esta función, para ser

precisos, en el cuerpo de la función *setNewDataToMSD*.

Para la ejecución distribuida esta función realiza la última inserción de *myStoredData* (conteniendo los datos para el último indicador) en el fichero de datos ya que esta última inserción no se va a llegar a realizar en la función *prepareforDist*. Para la ejecución no distribuida, esta función siempre realiza la Inserción de todo el data frame *myStoredData* cargado previamente con la función *addToMyStoredData*. Además de realizar las inserciones, esta función escribe información importante en el fichero de Log que indica que se puede haber descargado un fichero incorrecto o que no se ha descargado el correcto.

El detalle de la implementación de estos sub-procesos se adjunta en el Anexo V.

OpenData_Opt.r

La implementación de esta versión tiene un enfoque distinto a la versión anterior. Es una optimización más sencilla ya que no lleva control de cambios, por consiguiente el rendimiento también se optimiza. Esta versión sólo se usa para una versión distribuida y la estructura final de datos son ficheros similares a la versión anterior: [codigo_indicador]_myStoredData.csv.

Esta versión implementa todos los subprocesos indicados en la Tabla 8 (excepto el subproceso de descarga). Las siguientes secciones detallan cada uno de los subprocesos: "Carga de Estructura Temporal", "Verticalización y Formateo" e "Inserción en Estructura Final".

Carga de Estructura Temporal – R

Para esta versión se asume que el fichero ha sido ya descargado. La carga de estructura temporal se realiza de manera bastante sencilla usando la cláusula ***read.csv***.

Verticalización y Formateo – R

Este es el proceso más largo de esta versión. La idea es crear un data frame para cada hora. Estos data frames se irán cargando con los datos comunes, con el valor de medición de la columna correspondiente a cada hora. Luego, se unen todos estos data frames en uno único.

Inserción en Estructura Final – R

Esta sección detalla las instrucciones a seguir para almacenar los datos verticalizados en el fichero de estructura final de datos. El detalle de la implementación de estos sub-procesos se adjunta en el Anexo VI.

OpenData_Lib.r

Para la implementación de esta versión y de la siguiente se hace uso de funciones que optimizan el rendimiento, estas funciones se detallan en la Tabla 25.

Tabla 25. Funciones/Librerías externas usada para OpenData_Lib.r

Nº	Función	Librería	Descripción
1	getURL	rcurl	Permite realizar la descarga de un fichero
2	Melt	reshape2	permite hacer una Transposición de datos
3	filter	dplyr	Permite extraer datos de un data frame bajo ciertas condiciones.

Otra característica importante para esta versión es que la estructura final se modifica en cuanto a la fecha y hora. Antes esta se componía de 4 campos: *año, mes, día y hora*. (ver Tabla 7). En esta versión, la estructura final cuenta con un solo campo para la fecha, en la que se indica la fecha como cadena de caracteres que incluye el año, el mes, el día y la hora. La nueva estructura final es la mostrada en la Tabla 26.

Tabla 26. Nueva estructura final para OpenData_Lib.r

Columna	Descripción	Longitud
1	Comunidad Autónoma (28 : Comunidad de Madrid, 2 : Junta de Andalucía)	Delimitado por separador
2	Ciudad (79 : Comunidad de Madrid, 2 : Junta de Andalucía)	Delimitado por separador
3	Estación (Unión de Tabla 1 y Tabla 4)	Delimitado por separador
4	Indicador (Unión de Tabla 2 y Tabla 5)	Delimitado por separador
5	Fecha-Hora	Delimitado por separador
6	Valor de Medición	Delimitado por separador

Un fichero que contiene esta nueva estructura final, tendría la siguiente apariencia

mostrada en la Figura 20.

```
2,2,66,83,2016-03-17 21:00:00,5.73
2,2,67,83,2016-03-17 21:00:00,7.96
2,2,68,83,2016-03-17 21:00:00,2.37
2,2,75,83,2016-03-17 21:00:00,8.53
2,2,76,83,2016-03-17 21:00:00,5.6
2,2,80,83,2016-03-17 21:00:00,3.93
2,2,81,83,2016-03-17 21:00:00,9.96
2,2,82,83,2016-03-17 21:00:00,8.86
2,2,88,83,2016-03-17 21:00:00,12.88
2,2,89,83,2016-03-17 21:00:00,14.98
2,2,91,83,2016-03-17 21:00:00,7.07
2,2,92,83,2016-03-17 21:00:00,11.75
2,2,97,83,2016-03-17 21:00:00,14.98
2,2,103,83,2016-03-17 21:00:00,14.39
2,2,104,83,2016-03-17 21:00:00,16.83
2,2,130,83,2016-03-17 21:00:00,14.24
28,79,99,83,2016-03-17 01:00:00,9.1
28,79,4,83,2016-03-17 01:00:00,9.9
28,79,18,83,2016-03-17 01:00:00,10
28,79,24,83,2016-03-17 01:00:00,6.4
28,79,38,83,2016-03-17 01:00:00,12.9
28,79,54,83,2016-03-17 01:00:00,9.4
28,79,56,83,2016-03-17 01:00:00,8.4
28,79,57,83,2016-03-17 01:00:00,8.8
28,79,59,83,2016-03-17 01:00:00,7.5
28,79,99,83,2016-03-17 02:00:00,8.3
28,79,4,83,2016-03-17 02:00:00,9.9
28,79,18,83,2016-03-17 02:00:00,9.3
28,79,24,83,2016-03-17 02:00:00,4.8
28,79,38,83,2016-03-17 02:00:00,11.8
28,79,54,83,2016-03-17 02:00:00,8.7
28,79,56,83,2016-03-17 02:00:00,7.9
28,79,57,83,2016-03-17 02:00:00,7.7
28,79,59,83,2016-03-17 02:00:00,6.4
28,79,99,83,2016-03-17 03:00:00,7.9
28,79,4,83,2016-03-17 03:00:00,8.7
28,79,18,83,2016-03-17 03:00:00,9
28,79,24,83,2016-03-17 03:00:00,4.3
```

Figura 20. Nuevo fichero con la nueva estructura final para OpenData_Lib.r

Esta versión no lleva control de cambios y solo se aplica para una versión distribuida y la estructura final de datos son ficheros similares a la versión anterior:

Esta versión realiza también la descarga, por tanto se detallará cada uno de los 4 subprocesos: “Descarga”, “Carga de Estructura Temporal”, “Verticalización y Formateo” e “Inserción en Estructura Final”.

Descarga - R

El subproceso de descarga del fichero de datos se realiza en esta versión con ayuda de la función *getURL* de la librería *Rcurl*. Dado que en esta versión se implementa la descarga, ya se puede automatizar los 4 subprocesos dentro de un bucle. La implementación de la descarga se realiza en la misma sección de la “Carga de Estructura Temporal” por tanto esta se mostrará en esa sección.

Carga de Estructura Temporal - R

Al usar la función *getURL* para hacer la descarga del fichero de datos también se puede realizar la carga del contenido de este fichero en una estructura temporal.

Verticalización y Formateo – R

Esta sección detalla las instrucciones para realizar la verticalización de los datos, para ello se utiliza la función *melt* del paquete *reshape2* y *filter* del paquete *dplyr*. La función melt es la que realiza el proceso de verticalización. Esta función al recibir como entrada la Tabla 27, va a retornar la Tabla 28.

Tabla 27. Ejemplo de registro antes de realizar la función melt

Comaut	**Ciudad**	**Estaci**	**Parame**	**Ano**	**Mes**	**Dia**	**1**	**S1**	**2**	**S2**	**3**	**s3**
28	79	99	1	2016	01	13	6	V	7	V	8	N
28	79	99	2	2016	01	13	1	V	2	V	3	N

Tabla 28. Tabla generada después de la ejecución de la función melt

Comaut	Ciudad	Estaci	Parame	Ano	Mes	Dia	Horavl	Valorl
28	79	99	1	2016	01	13	1	6
28	79	99	1	2016	01	13	S1	V
28	79	99	1	2016	01	13	2	7
28	79	99	1	2016	01	13	S2	V
28	79	99	1	2016	01	13	3	8
28	79	99	1	2016	01	13	S3	N
28	79	99	2	2016	01	13	1	1
28	79	99	2	2016	01	13	S1	V
28	79	99	2	2016	01	13	2	7
28	79	99	2	2016	01	13	S2	V
28	79	99	2	2016	01	13	3	8
28	79	99	2	2016	01	13	S3	N

Inserción en Estructura Final – R

Esta sección detalla las instrucciones a seguir para almacenar los datos verticalizados en el fichero de estructura final de datos. El detalle de la implementación de estos sub-procesos se adjunta en el Anexo VII.

3.1.3 Rendimiento de las distintas versiones

Se ha hecho un análisis de rendimiento de todas las versiones implementadas. Este análisis consistió en evaluar las distintas implementaciones con dos ficheros: un fichero que contiene 2420 líneas y otro fichero que contiene 23721 líneas. A continuación se muestran los detalles de este análisis.

1. Primera Comparación

Los tiempos para una primera comparación se puede observar en la Tabla 29.

Tabla 29. Comparación de rendimiento de las versiones con un fichero de 540 Kb

Nº	Versión	Tecnolog.	Com. Aut.	Líneas Fichero	Tamaño Fichero	Dist	Control de Cambios	Campos de Fecha	Tiempo
1	Java-Oracle	Java-Oracle	Madrid	2420	540 kb	N/A	SI	4	00:00:03
2	OpenData_Dist.r	R	Madrid	2420	540 kb	SI	SI	4	00:00:12
3	OpenData_Dist.r	R	Madrid	2420	540 kb	NO	SI	4	00:00:12
4	OpenData_Opt.r	R	Madrid	2420	540 kb	SI	NO	4	00:00:01
5	OpenData_Lib.r	R (Librerías)	Madrid	2420	540 kb	SI	NO	1	00:00:01

En la Figura 21 se observa la comparativa mediante un gráfico de barras para la primera comparación.

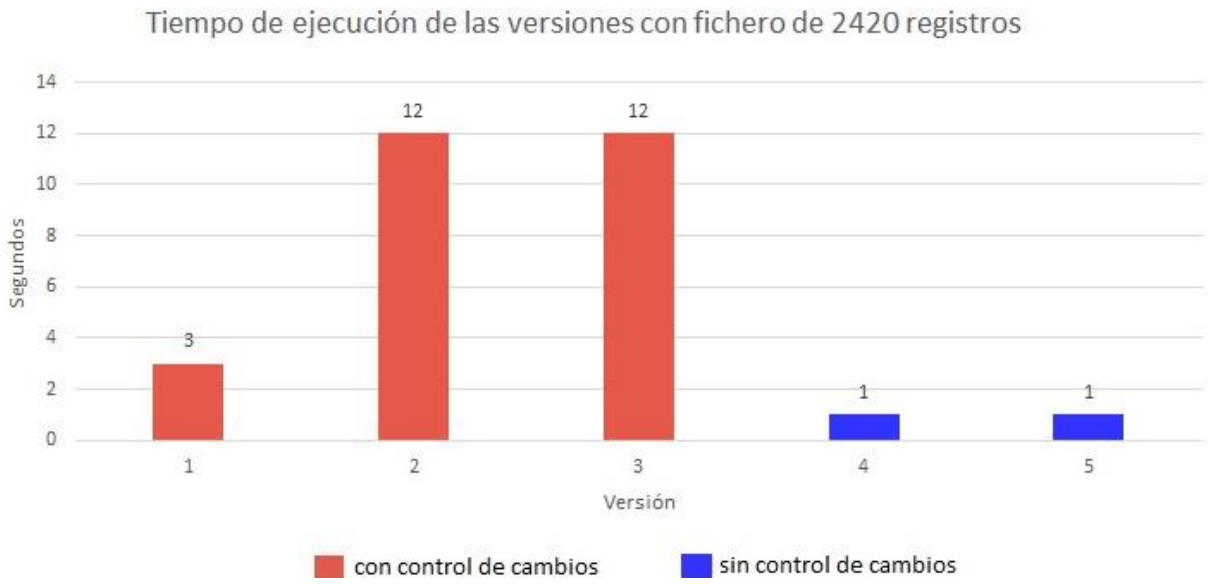


Figura 21. Gráfico de barras para la primera comparación

2. Segunda Comparación

Los tiempos para una segunda comparación se pueden observar en la Tabla 30.

Tabla 30. Comparación de rendimiento de las versiones con un fichero de 5.3 Mb

Nº	Versión	Tecnolog.	Com. Aut.	Líneas Fichero	Tamaño Fichero	Dist	Control de Cambios	Campos de Fecha	Tiempo
1	Java-Oracle	Java-Oracle	Madrid	23721	5.3 mb	N/A	SI	4	00:01:32
2	OpenData_Dist.r	R	Madrid	23721	5.3 mb	SI	SI	4	00:52:12
3	OpenData_Dist.r	R	Madrid	23721	5.3 mb	NO	SI	4	00:42:28
4	OpenData_Opt.r	R	Madrid	23721	5.3 mb	SI	NO	4	00:00:06
5	OpenData_Lib.r	R (Librerías)	Madrid	23721	5.3 mb	SI	NO	1	00:00:16

En la Figura 22 se observa la comparativa mediante un gráfico de barras para la segunda comparación.

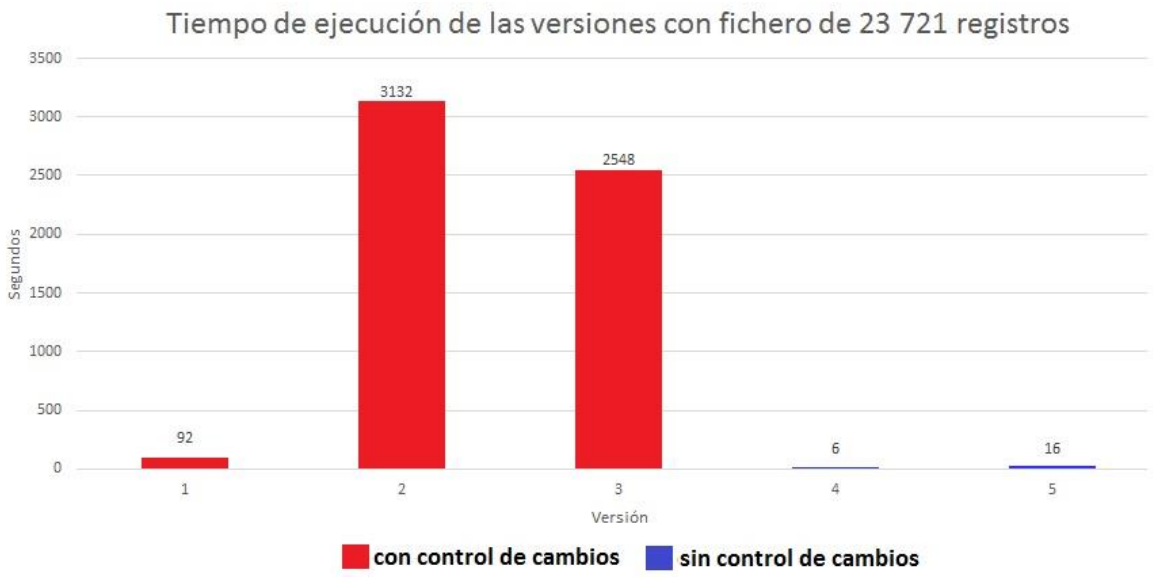


Figura 22. Gráfico de barras para la primera comparación

3.2 Integración del dataset de la Junta de Andalucía (Red Hidrosur)

En esta sección se detallará las instrucciones para integrar los datos de la Junta de Andalucía. Para conseguir esto, solo se ha implementado una única versión ya que el proceso de integración para estos datos no conlleva demasiada complejidad. Al igual que para la Comunidad de Madrid, la frecuencia de actualización de datos es de una hora.

Tal y como se puede ver en la Tabla 9, el proceso de integración para estos datos solo se ha implementado en lenguaje R y la implementación consta de 4 procesos: "Descarga", "Carga de Estructura Temporal", "Formateo" e "Inserción en Estructura Final".

3.2.1 Versión R

Es la única versión implementada para estos datos, y se usan también utilidades de R que ayudan a disminuir el código tales como la función *filter* de la librería *dplyr* y la función *getURL* de la librería *Rcurl*. Se implementan los 4 subprocesos, pero los dos primeros se encuentran solapados.

Descarga – R

El portal ofrece la posibilidad de realizar consultas de los datos mediante un formulario que se puede observar en la Figura 23 en el cual se especifica la estación y las fechas. La respuesta a estas consultas son los ficheros de datos necesarios para el proceso de integración. Es decir, nos permite descargar manualmente el fichero de datos.

Pero a diferencia de la Comunidad de Madrid, no cuenta con URL fijo que permita automatizar la descarga.

Como se ha mencionado ya previamente, para conseguir la automatización de la descarga de este fichero de datos, se aprovecha la forma en que el portal envía a su servidor los datos del formulario. Es decir, aprovechamos los parámetros que se envían en una URL al llamar al método *GET* de ese formulario.

Para aprovechar esto, se tiene que construir la URL en base a los parámetros sobre los cuales queremos consultar los datos. El detalle de la implementación de este sub-proceso se adjunta en el Anexo VIII.

Figura 23. Formulario de descarga de datos de la Red Hidrosur

Carga de Estructura Temporal – R

La "Descarga" como la "Carga de Estructura Temporal" se realiza en una sola sentencia tal y como se ha realizado la "Descarga" en la última versión R para el dataset de la Comunidad de Madrid.

Pero para este caso, la URL de descarga será la que se ha generado en la sección inmediatamente anterior.

El detalle de la implementación de este sub-proceso se adjunta en el Anexo IX.

Formateo – R

Este subproceso construye nuevas columnas a la estructura temporal a partir de las que ya tiene originalmente, para luego eliminar las originales y mantener las nuevas.

El detalle de la implementación de este sub-proceso se adjunta en el Anexo X.

Inserción en Estructura Final – R

Este subproceso se encarga de leer la estructura final que ya se tiene almacenada e incluir los datos nuevos que se han descargado.

El detalle de la implementación de este sub-proceso se adjunta en el Anexo XI.

Capítulo 4. Interface Gráfica

En este capítulo se muestra la implementación de una interface gráfica de usuario que permite mostrar los datos integrados en forma de gráficos para su análisis posterior por parte del usuario. Una vez que los datos se encuentran totalmente integrados, ya es posible crear servicios que sean comunes para todos los datasets que hayan pasado por el proceso de integración.

Tal y como se ha implementado el proceso de integración, esta interface gráfica también se ha implementado usando diferentes tecnologías. Hay que considerar que la tecnología utilizada para realizar la interface gráfica tiene que ser acorde con la tecnología usada en el proceso de integración, para ser precisos, tiene que ser acorde con la tecnología implementada para el último sub-proceso del capítulo de integración. Si se observa la Tabla 9, el último sub-proceso del proceso de integración es: "Inserción en Estructura Final", y de este sub-proceso se han implementado dos versiones con tecnologías distintas: Oracle y R.

Tabla 31. Versiones de implementación de la interface gráfica

Versión	**Programa Base**	**Módulos**			
		Formulario de Interface		**Estructura de Datos**	**Generación de Gráficos**
		Generación de Elementos de Formulario	**Extracción de datos para llenar Comboboxes**		
Java-Oracle-R	Java	N/A	N/A	Oracle	R
JavaScript-Java-R	N/A	JavaScript	Java	R	R
R	N/A	R	R	R	R

A continuación se detalla cada una de las versiones definidas en la Tabla 31.

4.1 Versión Java-Oracle-R

Esta versión no genera precisamente una interface gráfica ya que no permite que el usuario envíe sus parámetros para la generación de un gráfico. Se ha implementado simplemente como una continuación de la versión "Java-Oracle" del proceso de integración. Lo que se hace con esta versión, es generar un gráfico de un indicador y estación indicados en el código. Sin embargo, es importante mencionarla porque hace uso de una librería muy útil para la conexión

entre Oracle y R: “RJDBC”.

Debido a que ésta es una continuación de la versión “Java-Oracle” del proceso de integración, es la única versión que tiene programa base (ya que no es interface gráfica). Por tanto, el programa base, sigue estando en Java tal como en el proceso de integración.

Esta versión usa una clase Java: *RScriptGenerator* que se describe a continuación.

Clase RscriptGenerator: Esta clase tiene como objetivo el crear un script en R y se detalla en la Tabla 32. Dicho script genera un gráfico cuando es ejecutado, su respectivo diagrama de clase se puede observar en la Figura 24.

Tabla 32. Funciones de la clase RScriptGenerator

Nº	Método	Descripción
1	**Constructor**	El constructor sólo asigna valores recibidos como parámetros a variables de conexión a base de datos
2	**build_Script**	Este método genera un fichero script R. Este script es el que conectará R con Oracle y es donde se usa la librería ***RJDBC***. A su vez, este fichero es el que generará el gráfico en base a los indicadores recibidos por parámetro.
3	**get_RscriptFileName**	Este método devuelve el nombre del script R generado

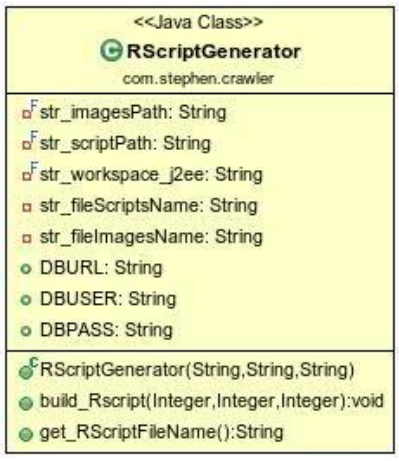


Figura 24. Diagrama de clases RScriptGenerator

El detalle de la implementación de esta versión de la interface gráfica se adjunta en el Anexo XII. Finalmente se genera el gráfico en la ruta indicada /ruta/de/imagenes/. Un ejemplo del resultado se puede observar en la Figura 25. Tal y como esta implementada esta versión, se generaría un gráfico para cada hora. En la Figura 26 se muestra un ejemplo del conjunto de

ficheros que se generaría.

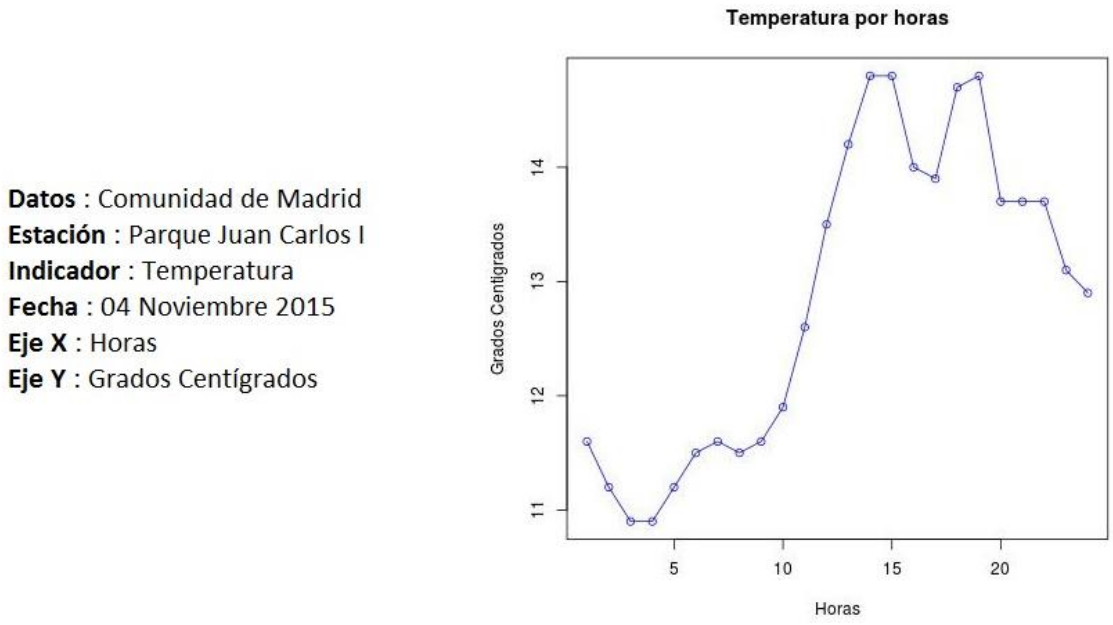


Figura 25. Gráfico generado por la versión Java-Oracle-R

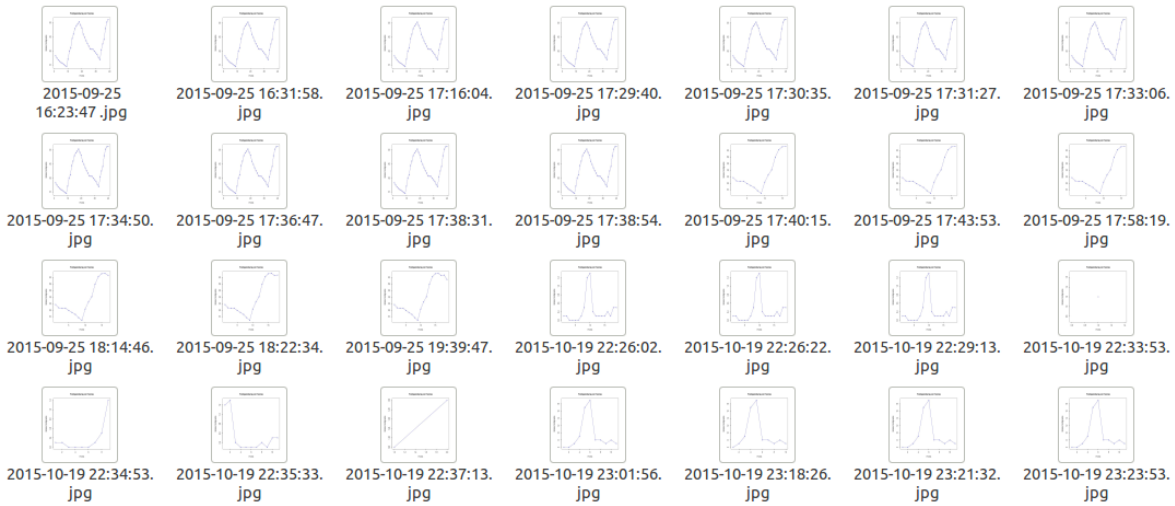


Figura 26. Gráficos generados por la versión Java-Oracle-R para cada hora

4.2 Versión JavaScript-Java-R

En esta sección se describe la implementación que he realizado de la interface gráfica donde el usuario puede enviar los parámetros que desee. Para la implementación de esta versión, se ha empleado JavaScript con librerías Jquery para la generación de elementos del formulario, se ha utilizado Java, en concreto *JSP* para permitir la ejecución de código R para la generación de gráficos.

Al formulario generado se le ha dado dinamismo, es decir los campos están anidados,

tanto los combos como los campos de fechas. Este dinamismo es útil porque evita que el usuario intente generar gráficos a partir de parámetros para las cuales no existen datos. Por ejemplo, si escogemos en el combo de Indicadores el indicador “Temperatura”, en el combo de estaciones solo se mostrarán aquellas estaciones que midan la temperatura. Para conseguir esto se han implementado algunas rutinas usando Ajax.

4.2.1 Elementos del Formulario

El formulario para la interface gráfica, es bastante sencillo para que el usuario pueda utilizarlo. Pero lleva cierta complejidad internamente ya que presenta dinamismo (Ajax) y tiene elementos que no son propios de HTML 5 que es la tecnología usada para el desarrollo de esta interface. La interface cuenta con 7 elementos y estos se muestran en la Tabla 33. Para implementar esta versión he creado los ficheros enumerados en la Tabla 34, utilizando tecnologías JSP (Java Server Pages) y R, según se indica en la Tabla 31.

Tabla 33. Elementos de la interface gráfica de la versión JavaScript-Java-R

Nº	**Etiqueta**	**Tipo**	**ID Elemento HTML**	**Descripción**
1	Parámetro	Combo	select_Parametro	Selector del indicador
2	Estación	Combo	select_Estacion	Selector de la estación
3	Desde	Text	id_Fecha_Desde	Indicador de la fecha Inicial
4	Hasta	Text	id_Fecha_Hasta	Indicador de la fecha Fin
5	Mantener Imágenes	CheckBox	id_mantener	Activado sobrepone los 4 últimos gráficos generados.
6	Generar	Boton	id_boton	Botón para generar el gráfico
7	N/A	N/A	Imagen	Elemento para mostrar el gráfico generado.

Tabla 34. Ficheros necesarios para la versión JavaScript-Java-R

Nº	**Nombre**	**Descripción**	**Tecnología**
1	Inicio.jsp	Levanta la interface gráfica.	JSP

2	ActualizarEstacionCombo.jsp	Actualiza datos en combo de Estaciones	JSP
3	ActualizarFechasInput.jsp	Actualiza datos en campo de Fechas	JSP
4	AlertNoFile.jsp	Valida que existan datos para el indicador seleccionado	JSP
5	ActualizarImagen.jsp	Genera el gráfico y lo ubica dentro de la interface gráfica.	JSP
6	GenericFunctions.r	Script R en el que se almacenan funciones de utilidad para la implementación (incluso se usa para la implementación de otras versiones)	R
7	WebInterface.r	Calcula, desde la estructura de datos, las opciones a mostrar en los combos como las fechas a mostrar en los campos	R
8	GenerarPlots.r	Genera el gráfico en base a los parámetros enviados	R
9	DataEst.tmp	Fichero temporal con datos de estaciones. Mecanismo para devolver la información desde R a Java.	
10	DataDte.tmp	Fichero temporal con datos de Fechas. Mecanismo para devolver la información desde R a Java.	
11	Mantener.tmp	Fichero temporal con la información de cada conjunto de parámetros enviados para generar el gráfico.	
12	Data.tmp	Fichero temporal con el nombre del gráfico a mostrar. Mecanismo para devolver la información desde R a Java.	

Para esta implementación se hará referencia a una clase Java: *GenericFunctions* que se describe a continuación.

Clase GenericFunctions. Esta clase contiene un método que se usará bastante para leer datos desde ficheros de datos. Este método es *getStaticData* y se detalla en la Tabla 35.

Una de las características de la interface gráfica es la capacidad de permitir al usuario elegir fechas (con la hora incluida) para establecer un rango de fechas. Aunque HTML5 tiene ya elementos que permiten ingresar datos de tiempo fecha-hora (*type="datetime",* estos no son soportados aún por muchos navegadores como Chrome, FireFox o Internet Explorer. Es por esto que se optó por utilizar una estructura de Jquery, concretamente el "DateTimePicker" que sí es compatible con estos navegadores. La Figura 29 muestra este elemento propio de esta interface gráfica.

El detalle de la implementación de esta versión de la interface gráfica se adjunta en el Anexo XIII. La interface gráfica implementada con esta versión se muestra en la Figura 30.

Tabla 35. Funciones del script GenericFunctions.r

Nº	Método	Descripción
1	**getStaticData**	Este método lee los datos de los ficheros de datos como lo pueden ser el fichero parametrosOD.csv (ver Figura 27) o el fichero estacionesOD.csv (ver Figura 28) y la información que recoge las almacena en una estructura ArrayList

```
01,Dióxido de Azufre
06,Monóxido de Carbono
07,Monóxido de Nitrógeno
08,Dióxido de Nitrógeno
09,Partículas < 2.5 µm
10,Partículas < 10 µm
12,Oxidos de Nitrógeno
14,Ozono
20,Toluneo
30,Benceno
35,Etilbenceno
37,Metaxileno
38,Paraxileno
39,Ortoxileno
42,Hidrocarburos Totales (hexano)
43,Hidrocarburos (Metano)
44,Hidrocarburos No Metánicos (hexano)
80,Radiación Ultravioleta
81,Velocidad de Viento
82,Dirección de Viento
83,Temperatura
86,Humedad Relativa
87,Presión
88,Radiación Solar
89,Precipitación
92,Lluvia ácida
```

Figura 27. Fichero parametrosOD.csv que contiene todos los indicadores

```
28,4,Pza. de Espana
28,8,Escuelas Aguirre
28,11,Av. Ramon y Cajal
28,16,Arturo Soria
28,17,Villaverde Alto
28,18,Farolillo
28,24,Casa de Campo
28,27,Barajas
28,35,Pza. del Carmen
28,36,Moratalaz
28,38,Cuatro Caminos
28,39,Barrio del Pilar
28,40,Vallecas
28,47,Mendez Alvaro
28,48,P. Castellana
28,49,Retiro
28,50,Pza. Castilla
28,54,Ensanche Vallecas
28,55,Urb. Embajada
28,56,Pza. Fdez. Ladreda
28,57,Sanchinarro
28,58,El Pardo
28,59,Parque Juan Carlos I
28,60,Tres Olivos
28,99,Media de la red
2,1,SIERRA MIJAS  (MA)
2,2,SIERRA DE LUNA (CA)
2,3,EMBALSE DE CHARCO REDONDO (CA)
2,6,LOS REALES (MA)
2,7,DEPOSITO DI-1 (CA)
2,16,EMBALSE DE LA CONCEPCION (MA)
2,22,MALAGA - PALACIO DE LA TINTA (MA)
2,27,RONDA (MA)
2,28,LAGUNA DE FUENTE PIEDRA (MA)
2,29,EMBALSE DEL  GUADALTEBA (MA)
2,34,AZUD DE PAREDONES (MA)
2,37,EMBALSE DE LA VINUELA (MA)
2,44,TORROX (MA)
```

Figura 28. Fichero estacionesOD.csv que contiene todas las estaciones

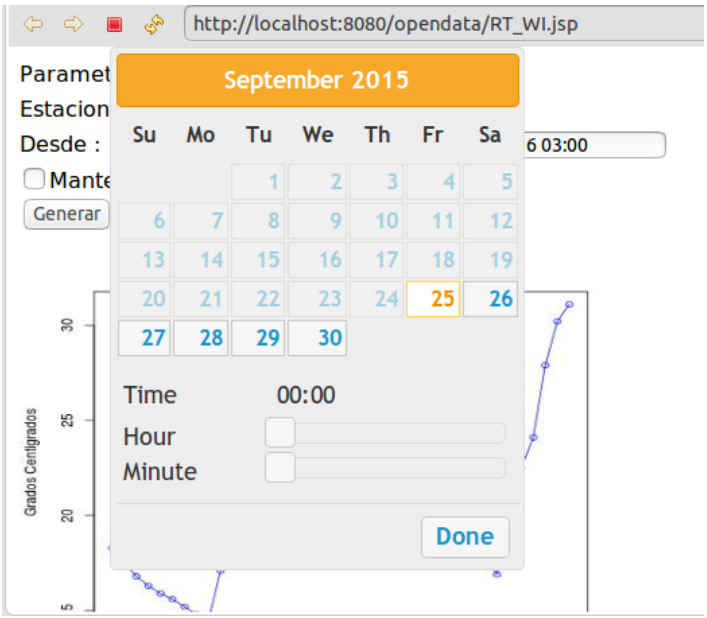


Figura 29. Elemento “DateTimePicker” para la interface gráfica: JavaScript-Java-R

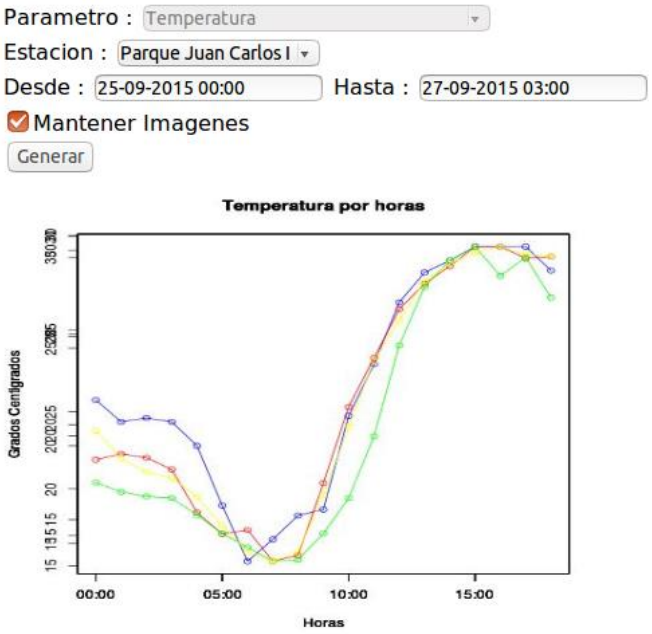


Figura 30. Interface generada con la versión JavaScript-Java-R

4.3 Versión R

Esta es la última versión que se ha realizado y está implementada en Open-Source, ya que se ha desarrollado totalmente en R. Esto ha sido posible gracias a una librería propia de R que permite realizar interfaces gráficas de manera bastante sencilla: *Shiny* [44].

Además, se ha usado otra librería R que permite optimizar el rendimiento de las consultas a la estructura de datos: *dplyr*. Por tanto es necesario que estas librerías estén instaladas en R.

Shiny es una librería que permite implementar interfaces gráficas con muy poco código. Sin embargo, es necesario seguir una estructura de código propia. Esta estructura consiste únicamente en elaborar dos ficheros para toda la implementación de la interface gráfica según están detallados en la Tabla 36.

Tabla 36. Módulos "Shiny" para la interface gráfica en versión R

Nº	Script	Descripción
1	server.r	Script de Servidor. Contiene instrucciones que el ordenador necesita para construir la aplicación.
2	ui.r	Script de Interface de Usuario. Controla el layout y apariencia de la aplicación.

Ambos ficheros tienen que estar en el mismo directorio sin importar el nombre del directorio. Para la implementación el directorio será: */ruta/de/mi/Shiny/app*

En el Anexo XIV se muestra los detalles de implementación de esta versión.

4.3.1 Ejecución

Para iniciar la interface gráfica de esta versión, se hace lo siguiente:

1. Entrar a la línea de comando de R.

```
> R
```

2. Cargar la librería Shiny.

```
> library(shiny)
```

3. Ejecutar la aplicación.

```
> runApp("/ruta/de/mi/Shiny/app")
```

La apariencia de la interface gráfica se puede ver en la Figura 31 y Figura 32.

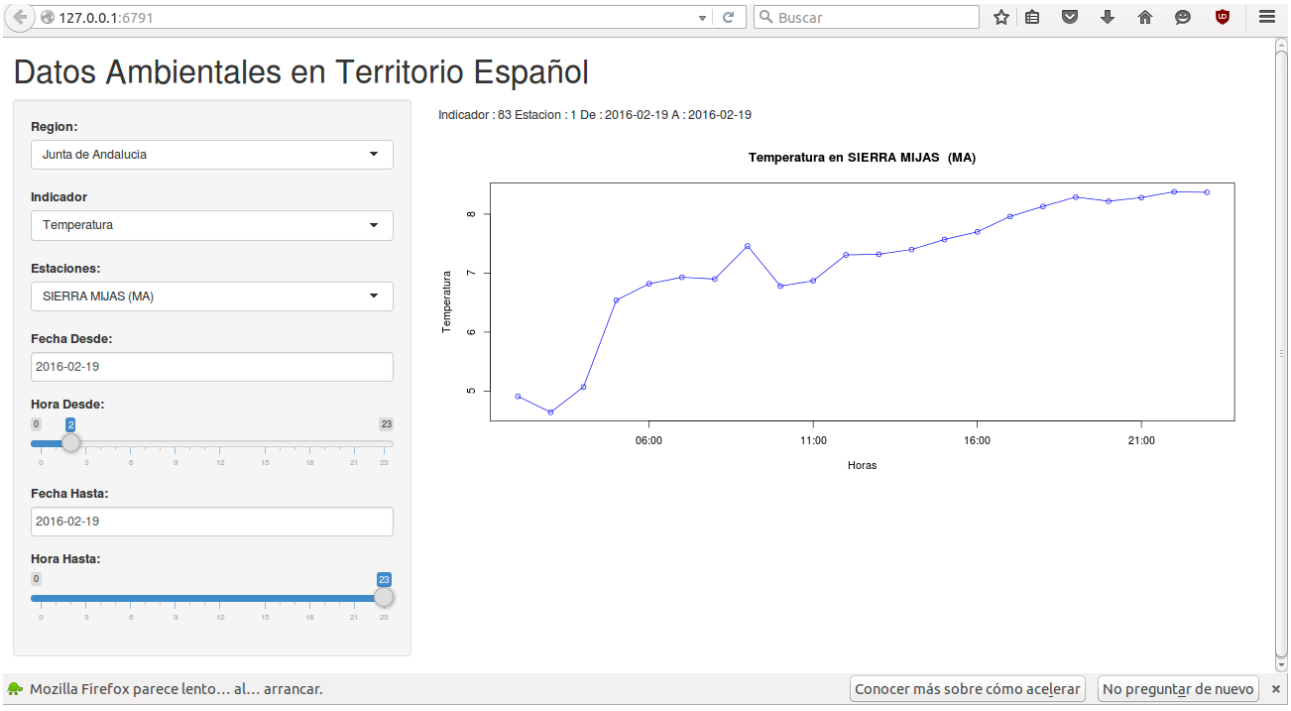


Figura 31. Interface gráfica generada con código R. Imagen 1

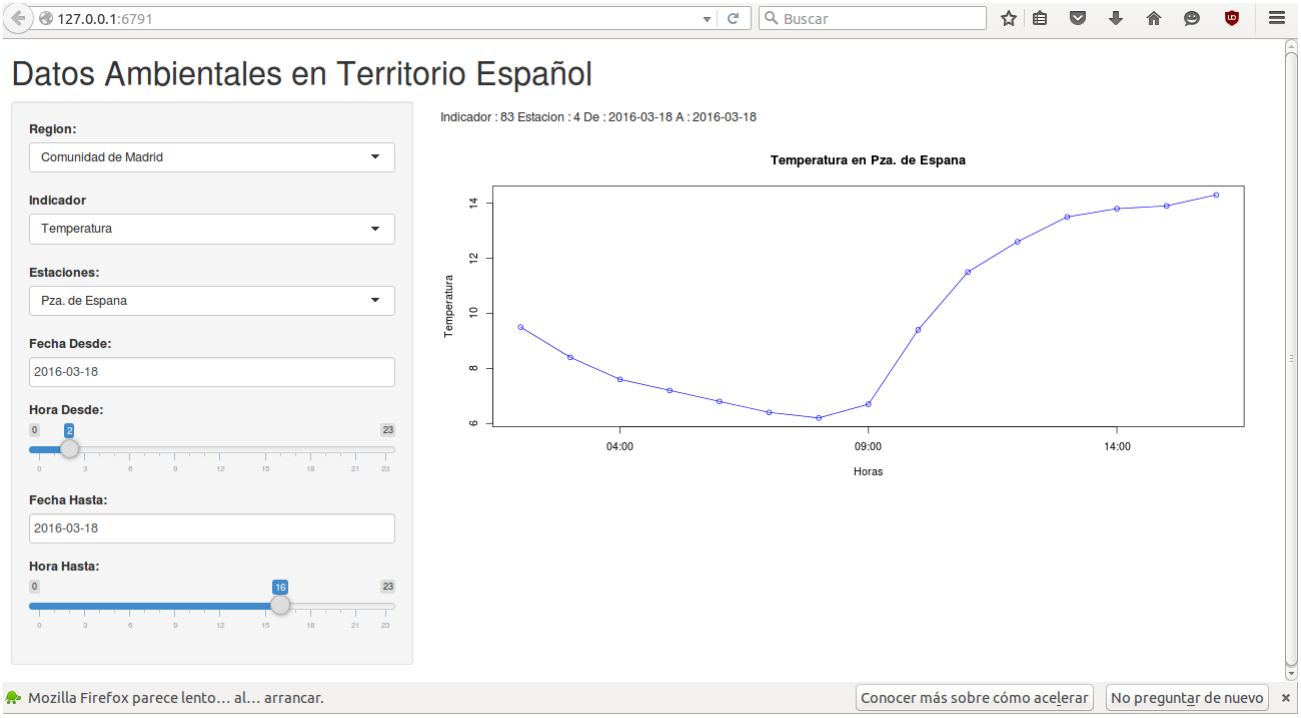


Figura 32. Interface gráfica generada con código R. Imagen 2

Capítulo 5. Conclusiones, Publicación de Resultados y Trabajo Futuro

En este Trabajo Fin de Máster se da respuesta a las cuestiones de investigación inicialmente planteadas. Se ha investigado cómo realizar una implementación de una aplicación que permite aprovechar los datasets en tiempo real relativos a datos medioambientales para facilitar la realización de análisis sobre los datos. Esta aplicación se ha implementado usando la herramienta Open-Source R mediante su lenguaje de programación. Se ha demostrado que con esta herramienta es posible elaborar un proceso de integración así como también una interface gráfica de manera bastante sencilla y con un rendimiento considerablemente bueno.

Como consecuencia de los avances de esta investigación se han ido publicando diversos trabajos en jornadas y congresos especializados. Los detalles de estos trabajos se muestran a continuación.

Congreso	VII Jornadas de Usuarios de R
Título	Procesamiento en R de un Open Data de Indicadores del Medio Ambiente en el Ayuntamiento de Madrid
Autores	Pavél Llamocca, Victoria López
Lugar	Salamanca
Fecha	6 de Noviembre del 2015
Resumen	Estas jornadas estaban enfocadas en el uso del lenguaje R en aplicaciones de uso sencillo y muy útil. Se presentaron las primeras versiones de la aplicación en las que se usó R y solo se contaba con la fuente de datos de la Comunidad de Madrid

Congreso	VIII Jornadas de Usuarios de R
Título	Procesamiento en R de un Open Data de Indicadores del Medio Ambiente en el Ayuntamiento de Madrid
Autores	Pavél Llamocca, Victoria López
Lugar	Albacete
Fecha	17 de Noviembre del 2016 (Aceptado para comunicación oral)
Resumen	Se presentará la aplicación implementada en su totalidad con lenguaje R y con el proceso de integración capaz de procesar datos de otra fuente de datos como la de la Junta de Andalucía.

Congreso	XXXVI Congreso Nacional de Estadística de Investigación Operativa (SEIO 2016)
Título	Integración de datos de medioambiente con R desde portales Open Data
Autores	Pavél Llamocca, Victoria López
Lugar	Toledo
Fecha	5-6 y 7 de Septiembre del 2016 (Aceptado para comunicación oral)
Resumen	El proyecto esta aceptado para su publicación. En esta comunicación se da más importancia a la parte de integración con R desde de la importancia de los datos en el muestreo, en el que se resaltarán los beneficios de lograr la integración de data sets de distintas fuentes. Aunque también se explicará el desarrollo y uso de una interface gráfica que permite al usuario analizar los datos de manera visual. Todo esto asociado a la versión desarrollada en R en su totalidad, disponible para seleccionar muestras de datos y realizar análisis comparativos.

Adicionalmente, en Abril de 2016 enviamos un artículo titulado: Data Integration of Environmetal Data al 18th International Conference on Data Mining, Big Data, Database and Data Engineering, y fue aceptado para su comunicación el 12 de Mayo de 2016 en Amsterdam. Sin embargo, a este congreso no pudimos asistir finalmente debido a problemas de diversa índole. No obstante el autor dispone de la carta de aceptación del trabajo.

Como trabajo futuro se propone la incorporación de nuevos datasets en territorio Español, e incluso a nivel internacional para poder deducir formatos normalizados sobre los que basar futuras analíticas sobre datos medioambientales. Se propone el uso de los nuevos conceptos *linked big data* [45][46] para simplificar los temas de descarga de datos y el análisis de las normativas nacionales e internacionales que vayan surgiendo para adaptar y mejorar los procedimientos de automatización de manera escalable. Esto implicaría la paralelización de los algoritmos en sistemas distribuidos y como consecuencia de todo ello la ampliación de los procesos automáticos de integración a diferentes áreas de interés.

Personalmente estoy considerando la posibilidad de continuar este trabajo hacia una tesis doctoral.

Referencias

[1]. Open Government Partnership (2015). The open government guide special Edition: Implementing the 2030 sustainable development agenda. Open Government Guide, 5-6. Recuperado de: http://www.opengovpartnership.org/sites/default/files/attachments/The%20Open%20Government%20Guide%20Special%20Edition.pdf

[2]. M. Harrison, T., Guerrero, Santiago (2011). Open Government and E-Government: Democratic Challenges from a Public Value Perspective, 2-5. Recuperado de : https://www.ctg.albany.edu/publications/journals/dgo2011_opengov/dgo2011_opengov.pdf

[3]. Westmore Nicola (2011). Transparency Opening Up Government, 3-3. Recuperado de: https://ico.org.uk/media/about-the-ico/documents/1042355/westmore-slideshow.pdf

[4]. European Commission (2015). Creating Value through Open Data: Study on the Impact of Re-use of Public Data Resources, 28-37. Recuperado de : https://www.europeandataportal.eu/sites/default/files/edp_creating_value_through_open_data_0.pdf

[5]. Nicandro Cruz-Rubio, C. (2014). Gobierno Abierto y Open Data. Actualidad Administrativa, 816-817. Recuperado de: https://funkziuni.files.wordpress.com/2014/07/aaj_7-8_2014_br.pdf

[6]. Darbishire, H. (2015, 16 de Octubre). Access Info presenta una queja formal sobre la falta de compromiso de España con el OGP. Access Info. Recuperado de : http://www.access-info.org/es/ogs-es/18735

[7]. Say M. (2015, 5 de Noviembre). Open data: barriers to be broken. UK Authority. Recuperado de : http://www.ukauthority.com/news/5743/open-data-barriers-to-be-broken#

[8]. The Open Book (s.f). Welcome to the Evolution of OPEN KNOWLEDGE timeline. Recuperado de : http://openbook.okfn.org/timeline/

[9]. Portal de datos Abierto de Cataluña (s.f). Open Data. Recuperado de: http://opendata.cloudbcn.cat/MULTI/es/what-is-open-data

[10]. Díaz R. (2013, 8 de Agosto). "Open Knowledge Foundation. La democratización de los datos". InfoLibre. Recuperado de: http://www.infolibre.es/noticias/medios/2013/08/08/open_knowledge_foundation_ong_del_open_data_6650_1027.html

[11]. James L. (2013, 3 de Octubre). "Open Knowledge International Blog. Defining Open Data". Recuperado de: http://blog.okfn.org/2013/10/03/defining-open-data/

[12]. Iglesias, C. (2013, 12 de Diciembre). European Public Sector Information Platform. A year of Open Data in the EMEA region. Recuperado de : http://www.epsiplatform.eu/sites/default/files/A%20year%20of%20Open%20Data%20in%20the%20EMEA%20region_0.pdf

[13]. Pollock R. (2007, 4 de Julio). Open Knowledge International Blog. The comprenhensive

knowledge archive network (CKAN) launched today. Recuperado de : http://blog.okfn.org/2007/07/04/the-comprehensive-knowledge-archive-network-ckan-launched-today/

[14]. W3C (s.f). Guía Breve de Web Semántica. Recuperado de: http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica

[15]. W3C (s.f). Wiki. Recuperado de: https://www.w3.org/wiki/SparqlEndpoints

[16]. Open Government Platform (s.f). Recuperado de : http://ogpl.gov.in/

[17]. Open Government Partnership (s.f). Recuperado de : http://www.opengovpartnership.org/es

[18]. Open Data Charter (s.f). Recuperado de : http://opendatacharter.net/

[19]. Global Open Data Initiative (s.f). Recuperado de: http://globalopendatainitiative.org/

[20]. Global Open Data Index (s.f). Recuperado de : http://index.okfn.org/

[21]. Open Data Research Network (s.f). Recuperado de: http://www.opendataresearch.org/

[22]. Open Data Institute (s.f). Partnership for Open Data. Recuperado de: https://theodi.org/odp4d

[23]. Open Data Institute (s.f). Global Development. Recuperado de : http://theodi.org/global-development-open-data

[24]. PowerData (2015, 18 de Mayo). "Qué significa integración de datos". PowerData. Recuperado de: http://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/405060/Qu-significa-la-integracin-de-datos

[25]. King T., "Data Integration and Data Cleansing Tools are at the Core of Big Data". (2014, 18 de Junio). Solutions Review. Recuperado de: http://solutionsreview.com/data-integration/data-integration-and-data-cleansing-tools-are-at-the-core-of-big-data/

[26]. Luna Dong X., "Big Data Integration". Presentado en 2013, 2013 IEEE 29th International Conference on Data Engineering (ICDE). Brisbane, QLD. pp. 1245-1248. ISBN. 978-1-4673-4909-3. Recuperado de : http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6544914

[27]. Smith M., "Big Data is Broker without Integration". (2013, 22 de Febrero). Venta Research. Recuperado de : https://marksmith.ventanaresearch.com/2013/02/22/big-data-is-broken-without-integration/

[28]. FlyData (s.f). "The 6 Challenges of Big Data Integration". FlyData. Recuperado de: https://www.flydata.com/the-6-challenges-of-big-data-integration/

[29]. Sinha Sudhi. "Making Big Data Work for Your Business". Publicado en 2014. ISBN 1783000996, 9781783000999. Packt Publishing Ltd, 2014

[30]. Luna Dong X., Srivastava D., "Big Data Integration: Synthesis Lectures on Data Management". Publicado en 2015. ISBN. 1627052240, 9781627052245. Morgan & Claypool Publishers, 2015

[31]. Wang D., Abdelzaher Tarek., Kaplan L. "Social Sensing: Building Reliable Systems on Unreliable Data". Publicado en 2015. ISBN. 0128011319, 9780128011317. Morgan Kaufmann.

[32]. Stonebraker M., Bruckner D., Ilyas Ihab F., Beskales G., Cherniack M., Zdonik S., Pagan A., Xu S., "Data Curation at Scale: The Data Tamer System", presentado en 2013, 6th Biennial Conference on Innovative Data Systems Research (CIDR '13), California, USA. Recuperado de:

http://cidrdb.org/cidr2013/Papers/CIDR13_Paper28.pdf.

[33]. Gubanov M., Stonebraker M., Bruckner D., "Text and Structured Data Fusion in Data Tamer at Scale". Recuperado de: http://www.mgubanov.com/docs/dt_icde.pdf

[34]. Kandel S., Paepcke A., Hellerstein J., Heer J. "Wrangler: Interactive Visual Specification of Data Transformation Script", presentado en 2011, Conference on Human Factors in Computing Systems, Vancouver, Canada. Recuperado de: http://vis.stanford.edu/files/2011-Wrangler-CHI.pdf

[35]. Petychakis M., Vasileiou O., Charilaos G. Spiros M. Psarras John. Journal of Theorical and Applied Electronic Commerce Research. (2013, 12 de Agosto). A State-of-the-Art Analysis of the Current Public Data Landscape from a Functional, Semantic and Technical Perspective. Recuperado de: http://www.scielo.cl/pdf/jtaer/v9n2/art04.pdf

[36]. Boletín Oficial del Estado (2013, 4 de Marzo). Norma Técnica de Interoperabilidad de Reutilización de recursos de la información. Recuperado de: http://www.boe.es/boe/dias/2013/03/04/pdfs/BOE-A-2013-2380.pdf.

[37]. Centro Tecnológico de la Información y Comunicación. (s.f). Recuperado de: https://www.fundacionctic.org/

[38]. Empresa Municipal de Transporte (s.f), Open Data. Servicio de Geolocalización. Recuperado de: http://opendata.emtmadrid.es/Servicios-web/GEO

[39]. Meijueiro, L. (2014, 26 de Marzo). Mapa actual de las iniciativas Open Data en España. Centro Tecnológico de la Información y la Comunicación (CTIC). Recuperado de : http://datos.fundacionctic.org/2014/03/mapa-actual-de-las-iniciativas-open-data-en-espana/

[40]. European Commission (2016, 15 de Febrero). Guidelines on Open Access to Scientific Publications and Research Data in Horizon 2020, 5-5. Recuperado de : http://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/hi/oa_pilot/h2020-hi-oa-pilot-guide_en.pdf

[41]. Comunidad de Madrid (s.f). Calidad del Aire en Tiempo Real. Recuperado de: http://www.mambiente.munimadrid.es/opendata/horario.txt.

[42]. Consejería de Medio Ambiente y Ordenación del Territorio, Junta de Andalucía (s.f). Red Hidrosur. Recuperado de: http://www.redhidrosurmedioambiente.es/

[43]. Rouse M. (s.f). Search SOA. Crawler. Recuperado de: http://searchsoa.techtarget.com/definition/crawler

[44]. RStudio (s.f) Shiny by RStudio A web application framework for R. Recuperado de : http://shiny.rstudio.com/

[45]. Pascal Hitzler, Krzysztof Janowicz. 2012. "Linked Data, Big Data and the 4th Paradigm", Editorial, Semantic web IOS Press. Recuperado de : http://www.semantic-web-journal.net/system/files/swj488.pdf

[46]. Shafani M., Bart H., Afsarmanesh. (s.f). "Exploring (bio)medical knowledge using Linked Data". BioMed Xplorer. Recuperado de : https://www.researchgate.net/publication/297403384_BioMed_Xplorer_-

_Exploring_BioMedical_Knowledge_using_Linked_Data

Anexos

Anexo I

Proceso: Integración datos de la Comunidad de Madrid

Versión: Java-Oracle

Sub-proceso: Descarga-java

DETALLE DE IMPLEMENTACIÓN

La implementación se basa en los siguientes pasos:

1. Instanciar la clase *CrawlerConstructorArbol* que luego usaremos para encontrar la URL en la que se encuentra el dataset. Para efectos de rendimiento, la URL base será ya aquella que referencie a la página web en la que se encuentre el link de descarga del dataset, es decir, la variable *str_dataset* contendrá la siguiente cadena.

```
String str_dataset =
"http://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=
41e01e007c9db410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCR
D";

CrawlerConstructorArbol spider = new CrawlerConstructorArbol(str_dataset);
```

2. Construir el árbol con solo un nivel de profundidad ya que la URL base es aquella que contiene el enlace de descarga del dataset.

```
spider.construirArbol(1);
```

3. Una vez el árbol este construido, obtenemos aquella URL que contenga el patrón de búsqueda que lo enviaremos como parámetro, en este caso “calidad-aire-tiempo-real.txt”.

```
String url = spider.get_urlContainedWord("calidad-aire-tiempo-real.txt");
```

4. Instanciamos la clase *WebDownloader* que servirá posteriormente para descargar el dataset y compararlo con el último dataset descargado previamente.

```
/*
 * Se descargará el contenido web del parámetro url
 * y será guardado con el nombre horario.txt
```

```
 */
WebDownloader wd = new WebDownloader (url,"horario.txt");
```

5. Descargamos el dataset, pero por otra parte también obtenemos el último fichero descargado previamente.

```
try
{
     wd.DownloadWeb();

     newestFile = wd.lastFileModified(str_rutaDescarga);
```

6. En caso que ambos ficheros sean iguales, el dataset no ha sido actualizado aún y se vuelve a esperar una hora. Esto se realiza por no cargar dos veces el mismo fichero en caso que por algún motivo la Comunidad de Madrid no haya actualizado el dataset en su web.

```
     try {
           /*
            * Comparamos ambos ficheros
            */
            cmp=wd.compareFile(newestFile.getPath(),wd.getFilePath());

           /*
            * Si ambos ficheros son iguales, entonces
            * el Data Set no ha sido actualizado
            * Por tanto terminamos esta vuelta del loop
            */

            if(cmp==1)     continue;

     } catch (Exception e) {
          e.printStackTrace();
     }

} catch (IOException e) {
     e.printStackTrace();
     System.exit(0);
}
```

Anexo II

Proceso: Integración datos de la Comunidad de Madrid

Versión: Java-Oracle

Sub-proceso: Carga de Estructura Temporal - Oracle

DETALLE DE IMPLEMENTACIÓN

La implementación se basa en los siguientes pasos:

1. Instanciar un objeto de la clase *DBConnector* y establecer la conexión

```
DBConnector connector = new DBConnector("opendata","usuario","password");

try {
      connector.Connect();

} catch (SQLException e) {
      e.printStackTrace();
      stem.exit(0);
}
```

2. Ejecutar el procedimiento almacenado *p_loadData* del paquete *Pkg_OpenData*, para que realice la carga del fichero de datos. El nombre del fichero se lo indicamos como parámetro.

```
connector.Exec_StoreProcedureOneStrPar("Pkg_opendata.p_loadData",
                                       "horario.txt");
```

Anexo III

Proceso: Integración datos de la Comunidad de Madrid

Versión: Java-Oracle

Sub-proceso: Verticalización y Formateo - Oracle

DETALLE DE IMPLEMENTACIÓN

La implementación se basa en los siguientes pasos:

1. Se invoca la función *Exec_StoreProcedureNoPars* para que ejecute la función *p_VerticalizarDatos* del paquete *Pkg_OpenData***.**

```
connector.Exec_StoreProcedureNoPars("Pkg_opendata.p_VerticalizarDatos");
```

Ejecutada esta instrucción, se lanzan las sentencias necesarias para que se cargue la estructura final.

Anexo IV

Proceso: Integración datos de la Comunidad de Madrid

Versión: Java-Oracle

Sub-proceso: Inserción en Estructura Final - Oracle

DETALLE DE IMPLEMENTACIÓN

La implementación se basa en los siguientes pasos:

1. Se ejecuta la función *Exec_StoreProcedureNoPars* para que ejecute el procedimiento almacenado *p_BorrarTabla* del paquete *Pkg_OpenData*.

```
connector.Exec_StoreProcedureNoPars("Pkg_opendata.p_BorrarTabla");
```

2. Se desconecta de la base de datos **open data**

```
try {
      connector.Disconnect();
} catch (SQLException e) {
      e.printStackTrace();
}
```

El programa base en Java de esta versión continua en la versión Java-Oracle-R de la interface gráfica, ya que esa versión es una continuación de la implementación de la versión Java-Oracle del proceso de integración.

Anexo V

Proceso: Integración datos de la Comunidad de Madrid

Versión: R (OpenData_Dist.r)

Sub-procesos: Carga de Estructura Temporal-R, Verticalización y Formateo –R e Inserción en Estructura Final-R.

DETALLE DE IMPLEMENTACIÓN

La implementación para esta versión se basa en los siguientes pasos:

1. Se especifica qué tipo de ejecución se desea en la variable booleana *bDist*, TRUE es para la ejecución distribuida y FALSE para la no distribuida

```
############################################################
#Indicador si se va a lanzar de manera distribuida o no    #
############################################################
bDist<<-TRUE
```

2. Se ejecuta la función ***read.csv*** para cargar el fichero descargado que es un fichero csv y almacenamos su contenido en el data frame global *myCurrentData*.

```
############################################################
# Cargamos en una estructura Temporal los datos descargados #
############################################################
myCurrentData<<-read.csv("/myOpenData/path/horario.txt",header=FALSE,stringsAsFactors=FALSE)
```

La implementación de los siguientes subprocesos "Verticalización y Formateo" e "Inserción en estructura final" se realiza dentro de la función *setNewDataToMSD*. Se detalla esta función en la Tabla 22.

3. Se ejecuta la función *setNewDataToMSD* al mismo tiempo que se escribe una línea en el log ya que esta función devuelve una cadena con la cantidad de registros integrados y con la cantidad de registros corregidos.

```
cat("Resultado Proceso R:
",setNewDataToMSD(),"\n",file="/myOpenData/path/logOD.txt",append=TRUE)
```

Esta sería una de las maneras de integrar los datos de forma "distribuida" o "no distribuida" con tecnología R.

Anexo VI

Proceso: Integración datos de la Comunidad de Madrid

Versión: R (OpenData_Opt.r)

Sub-procesos: Carga de Estructura Temporal-R, Verticalización y Formateo –R e Inserción en Estructura Final-R.

DETALLE DE IMPLEMENTACIÓN

Las instrucciones para implementar esta versión son:

1. Se usa la cláusula *read.csv* para cargar el fichero el cual volcará los datos a un data frame *myCurrentData*.

```
#################################################################
# Se carga el fichero descargado a un data frame myCurrentData  #
#################################################################
strFilePath<-"/myOpenData/path/horario.txt"
myCurrentData<-read.csv(strFilePath,header=FALSE,stringsAsFactors=FALSE)
```

2. Crear un data frame para cada hora *myCurrentData.HXX* En este data frame, añadimos las 7 columnas que son comunes y añadimos el valor de medición para la hora correspondiente. Esto se realiza únicamente si el valor esta verificado. Estos data frames tendrán la misma apariencia que la estructura final.

```
#################################################################################
# Se crea un data frame para cada hora en el cual solo se guarda los que        #
# estén verificados. Solo se consideran las 7 primeras columnas que son         #
# comunes, y además se coge la columna 10 que corresponde al valor de           #
# medición de la hora 1                                                         #
#################################################################################
myCurrentData.H01<-myCurrentData[myCurrentData[,11]=='V',c(1,2,3,4,7,8,9,10)]
```

3. Se asigna una cabecera y se añade una columna nueva que será la hora.

```
############################
# Se asigna una cabecera   #
############################
names(myCurrentData.H01) <- c("comaut","ciudad","estaci","parame","ano","mes","dia","valorl")
#########################################################
# Se añade el campo de la hora y se reordenan las columnas #
#########################################################
myCurrentData.H01$horavl <- 0
myCurrentData.H01<-
myCurrentData.H01[,c("comaut","ciudad","estaci","parame","ano","mes","dia","horavl","valorl")]
```

4. Se realiza lo anterior para todas las horas

```
#####################################################################
# Se repite lo anterior para cada hora.                             #
```

```
# Observar que la columna a evaluar si es Verificado o No, cambia,    #
# ahora es 13, y la columna con el valor de medición es ahora el 12   #
#########################################################################
myCurrentData.H02<-myCurrentData[myCurrentData[,13]=='V',c(1,2,3,4,7,8,9,12)]
names(myCurrentData.H02) <- c("comaut","ciudad","estaci","parame","ano","mes","dia","valorl")
myCurrentData.H02$horavl <- 1
myCurrentData.H02<-
myCurrentData.H02[,c("comaut","ciudad","estaci","parame","ano","mes","dia","horavl","valorl")]

###################
# Todas las horas #
###################

###################
# Hora 24         #
###################
myCurrentData.H24<-myCurrentData[myCurrentData[,57]=='V',c(1,2,3,4,7,8,9,56)]
names(myCurrentData.H24) <- c("comaut","ciudad","estaci","parame","ano","mes","dia","valorl")
myCurrentData.H24$horavl <- 23
myCurrentData.H24<-
myCurrentData.H24[,c("comaut","ciudad","estaci","parame","ano","mes","dia","horavl","valorl")]
```

5. Se unen los 24 data frames en un único data frame : *myToDayData*

```
#############################################################
# Unir verticalmente todos los data frames en myToDayData #
#############################################################
myToDayData<-
rbind(myCurrentData.H01,myCurrentData.H02,myCurrentData.H03,myCurrentData.H04,myCurrentData.H0
5,myCurrentData.H06,

myCurrentData.H07,myCurrentData.H08,myCurrentData.H09,myCurrentData.H10,myCurrentData.H11,myCu
rrentData.H12,

myCurrentData.H13,myCurrentData.H14,myCurrentData.H15,myCurrentData.H16,myCurrentData.H17,myCu
rrentData.H18,

myCurrentData.H19,myCurrentData.H20,myCurrentData.H21,myCurrentData.H22,myCurrentData.H23,myCu
rrentData.H24)
```

6. Se extrae los indicadores únicos de *myStoredData* en *indicadores.unicos*

```
#########################################################################
# Se extrae los indicadores únicos que tenemos en el fichero descargado  #
#########################################################################
indicadores.unicos<-unique(myCurrentData[,4])
```

7. Se guarda el año, mes y día de *myCurrentData*

```
#########################################################################
# Del primer registro del dataframe guardamos el año, mes y día, porque  #
# en los demás registros serán los mismo                                 #
#########################################################################
ano<-myCurrentData[1,7]
mes<-myCurrentData[1,8]
dia<-myCurrentData[1,9]
```

8. Se recorre cada indicador con un loop para ir cargando en *myStoredData* cada estructura final de cada indicador ya almacenada.

```
#####################################################################
# Se recorre con un loop los indicadores que están en myCurrentData  #
#####################################################################
for ( indicador in indicadores.unicos )
{
```

```
  ########################################################################
  # Se carga en myStoredData los datos del indicador del loop que se     #
  # encuentran ya en el fichero de Estructura Final                      #
  ########################################################################
  strStoredDataPath<-paste(strGlobalPath,indicador,"_myStoredData.csv",sep="")
  if (file.exists(strStoredDataPath))
  {
    myStoredData<-read.csv(strStoredDataPath,header=TRUE,fill=FALSE,strip.white=TRUE)
  }
```

9. De *myStoredData* se borran los datos de la fecha de *myCurrentData* para luego añadir todos los datos que están en *myToDayData*

```
  ##############################################################################################
  # Debido a que en myCurrentData vienen todas las horas anteriores a la hora                  #
  # actual de un día en particular, de myStoredData se borran las horas anteriores para        #
  # reemplazarlas posteriormente al cargar myToDayData y así cargar nuevamente los             #
  # datos del mismo día por si alguno ha sido modificado                                       #
  ##############################################################################################
  myStoredData<-myStoredData[!(myStoredData[,5]==ano & myStoredData[,6]==mes &
myStoredData[,7]==dia),]

  ##############################################################################################
  # Se añade a myStoredData los datos de myTodayData que sean del indicador actual del loop y  #
  # de la misma fecha de myCurrentData (variables ano,mes dia)                                 #
  ##############################################################################################
  myStoredData<-rbind(myStoredData,myToDayData[myToDayData[,4]==indicador &
myToDayData[,5]==ano & myToDayData[,6]==mes & myToDayData[,7]==dia,])
```

10. Finalmente se vuelcan los datos al fichero que es la estructura final.

```
  ############################################################################
  # Volcar los datos de myStoredData en el fichero de Estructura Final de datos #
  ############################################################################
  write.csv(myStoredData, file=strStoredDataPath,row.names=FALSE)

}
```

Anexo VII

Proceso: Integración datos de la Comunidad de Madrid

Versión: R (OpenData_Lib.r)

Sub-procesos: Descarga-R, Carga de Estructura Temporal-R, Verticalización y Formateo –R e Inserción en Estructura Final-R.

DETALLE DE IMPLEMENTACIÓN

Las instrucciones para implementar esta versión son:

1. Antes de empezar la implementación, se tiene que invocar las librerías antes mencionadas.

```
library(reshape2) # melt
library(dplyr) #filter
library(bitops) # necesario con dplyr
library(RCurl) # getURL
```

2. Debido a que en esta versión, se implementa la descarga, se puede automatizar los 4 subprocesos en un loop. Dentro del loop se realiza la descarga del fichero con la función *getURL* indicando la URL desde donde se descarga el fichero de datos. Al usar esta función se está también ya almacenando el contenido del fichero en el data frame *myCurrentData*.

```
while(1==1){
  ################################################################################
  # Se realiza la descarga indicando la URL de la fuente de datos. La función  #
  # para descargar es getURL de la librería RCurl.                             #
  # El fichero descargado se almacena en su forma original en el data frame    #
  # myCurrentData                                                              #
  ################################################################################
  myCurrentData<-
read.csv(textConnection(getURL("http://www.mambiente.munimadrid.es/opendata/horario.txt")),
header = FALSE, skip = 1, stringsAsFactors = FALSE, sep = ",")
```

3. Se asigna cabeceras a *myCurrentData*. A cada columna correspondiente al valor de medición de una hora en concreto, se le asigna una cabecera que es la misma hora: 1, 2, 3, .. 0 (0 en lugar de 24). Y a cada columna correspondiente a la verificación de cada valor de medición de cada hora, se le asigna una cabecera de la forma: sX; es decir: s1, s2, s3, s0.

```
  ##############################################################################
  # Se asigna cabeceras correspondientes a las horas. Estas cabeceras pasarán  #
  # posteriormente a ser valores cuando usemos el "melt".                      #
  # Observar que se empieza con la hora 1, y terminamos con la hora 0          #
  # (no la hora 24 como la comunidad de Madrid nos indica).                    #
  # A estos registros de hora 0 luego les cambiaremos la fecha a un día más.   #
  ##############################################################################
  names(myCurrentData) <-
c("comaut","ciudad","estaci","parame","tecana","perana","ano","mes","dia",
"1","s1","2","s2","3","s3","4","s4","5","s5","6","s6","7","s7","8","s8","9","s9","10","s10","1
1","s11","12","s12","13","s13","14","s14",
"15","s15","16","s16","17","s17","18","s18","19","s19","20","s20","21","s21","22","s22","23","
s23","0","s0")
```

4. Se eliminan las columnas que no van a ser necesarias a lo largo del proceso: “Técnica Analítica” y “Periodo de Análisis”.

```
  ###############################################
  # Se eliminan las columnas innecesarias:      #
  # Técnica analítica y Periodo de Análisis     #
  ###############################################
  myCurrentData<-myCurrentData[,-(5:6)]
```

5. Se realiza la transposición (Verticalización) con la función *melt*. Esta función realiza una transposición de cada una de las columnas que no son consideradas clave (id) manteniendo estas columnas clave como comunes a todos los registros. La cabecera original de cada campo transpuesto se convertirá en valores de una nueva columna cuyo nombre se define con el parámetro *variable.name*. Y los valores originales de cada campo transpuesto se convertirá en valores de una nueva columna cuyo nombre se define con el parámetro *value.name*

```
  ############################################################
  # Función melt - librería reshape2. Realiza una Transposición #
  ############################################################
  myCurrentData<-melt(myCurrentData,id=1:7,variable.name="horavl",value.name="valorl")
```

6. Después de realizado el melt, solo interesa mantener en *myCurrentData* aquellos registros que indican los valores de medición, es decir, aquellos en los que la columna *valorl* sea distinta de “V” o “N” o cualquier otro valor que pueda venir en esa columna

```
  ####################################################################
```

```
# Se excluyen los registros correspondientes a la columna verificación #
#########################################################################
myCurrentData<-filter(myCurrentData,valorl!="N" & valorl!="V" & valorl!="M" & valorl!="C")
```

7. Se añade la columna fechal a *myCurrentData*.

```
#######################################################
# Se construye la columna fecha en myCurrentData      #
# con los datos del primer registro de myCurrentData  #
#######################################################
ano<-myCurrentData[1,5]
mes<-myCurrentData[1,6]
dia<-myCurrentData[1,7]

myCurrentData$fechal<-as.POSIXct(paste(myCurrentData$ano,"-",myCurrentData$mes,"-
",myCurrentData$dia," ",myCurrentData$horavl,":00:00",sep=""))
```

8. Se guarda en *currentdate* la fecha de *myCurrentData* y se añade un día a las horas 00:00.

```
#######################################################
# Se guarda en currentdate la primera hora (00:00) de #
# la fecha de myCurrentData                           #
#######################################################
currentdate<-as.POSIXct(paste(ano,"-",mes,"-",dia," 00:00:00",sep=""))

###############################################
# PARA LAS HORAS 24                           #
# a los registros de hora 00:00, se les       #
# suma un día                                 #
###############################################
dateplusone<-currentdate+(60*60*24)
myCurrentData[as.numeric(format(as.POSIXct(myCurrentData$fechal),"%H"))==0,10]<-dateplusone
```

9. Se convierte la columna fechal de *myCurrentData* en una cadena de caracteres

```
#####################################
# Se convierte la columna de fecha  #
# de tipo fecha a tipo carácter     #
#####################################
myCurrentData[,10]<-as.character(myCurrentData[,10])
```

10. Se eliminan los registros posteriores a la hora actual ya que no estarán verificados y vendrán con valores de medición a 0.

```
#####################################################
# Se excluye los registros correspondientes a horas #
# posteriores a la hora actual porque no interesan  #
#####################################################
myCurrentData<-filter(myCurrentData,as.POSIXct(myCurrentData$fechal)<=Sys.time())
```

11. Dado que ya tenemos una nueva columna fechal, no es necesario mantener las columnas de *año*, *mes* y *día*, por lo tanto se eliminan

```
#################################################
# Se borran los campos año, mes y dia ya que tenemos #
# una nueva columna de fecha y reordenamos          #
#################################################
```

```
  myCurrentData<-myCurrentData[,-(5:8)]
  myCurrentData<-myCurrentData[c("comaut","ciudad","estaci","parame","fechal","valorl")]
```

12. Se extrae los indicadores únicos de *myStoredData* en *indicadores.unicos*.

```
#########################################################################
# Se extrae los indicadores únicos que tenemos en el fichero descargado  #
#########################################################################
indicadores.unicos<-unique(myCurrentData[,4])
```

13. Se recorre con un loop cada uno de los indicadores únicos y se carga cada estructura final correspondiente a cada indicador en el data frame *myStoredData.*

```
  ############################################################
  # Ruta donde se encuentran los ficheros de Estructura Final #
  ############################################################
  strDataPath<-"/myOpenData/path/data/datav2/"

  ######################################################################
  # Se recorre con un loop los indicadores que están en myCurrentData  #
  ######################################################################
  for ( indicador in indicadores.unicos )
  {
    ##########################################################################
    # Se carga en myStoredData los datos del indicador del loop que se      #
    # encuentran ya en el fichero de Estructura Final                       #
    ##########################################################################
    strStoredDataPath<-paste(strDataPath,indicador,"_myStoredDatav2.csv",sep="")
    if (file.exists(strStoredDataPath))
    {
      myStoredData<-read.csv(strStoredDataPath,header=TRUE,fill=FALSE,strip.white=TRUE)
    }
```

14. Se elimina de *myStoredData* aquellos datos del día actual (*currentdate*) desde las 00:00 horas en adelante, porque se encuentran en *myCurrentData* y serán cargados nuevamente posteriormente. Esto permite siempre mantener los últimos datos así hayan sufrido cambios.

```
    ##############################################################################
    # De myStoredData se excluye aquellos cuya fecha-hora sea mayor que la hora 0   #
    # de la fecha de myCurrentData, es decir, currentdate.                       #
    ##############################################################################
    myStoredData<-myStoredData[!(myStoredData[,1]==28 & myStoredData[,2]==79 &
as.numeric(format(currentdate,"%Y%m%d%H")) <
as.numeric(format(as.POSIXct(myStoredData[,5],format="%Y-%m-%d %H:%M:%S"),"%Y%m%d%H"))),]
```

15. Se añade a *myStoredData* los datos de *myCurrentData* que contendrá los datos de todas las horas de la fecha actual (*currentdate*).

```
    ##########################################################################################
    # Se añade a myStoredData los datos de myStoredData que son del indicador actual del #
    # loop y de la misma fecha de myCurrentData (variables ano,mes dia)                    #
    ##########################################################################################
    myStoredData<-rbind(myStoredData,myCurrentData[myCurrentData[,4]==indicador,])
```

16. Se vuelca los datos de *myStoredData* en el fichero de la estructura final y se cierra el loop de indicadores.

```
    ##############################################################################
    # Volcar los datos de myStoredData en el fichero de Estructura Final de datos #
    ##############################################################################
    write.csv(myStoredData, file=strStoredDataPath,row.names=FALSE,quote=FALSE)

  }
```

17. Se duerme el proceso para repetir todas las instrucciones dentro de una hora. Y se cierra el bucle.

```
  Sys.sleep(3600)

}
```

Estas serían todas las instrucciones para realizar el proceso de integración con R y librerías que ayudan a la optimización.

Anexo VIII

Proceso: Integración datos de la Junta de Andalucia (Red Hidrosur)

Versión: R

Sub-proceso: Descarga-R

DETALLE DE IMPLEMENTACIÓN

Las instrucciones para construir la URL para descargar el fichero son:

1. Antes de empezar, se tiene que cargar las librerías que van a ser usadas y comenzar un bucle para automatizar la descarga con una frecuencia de una hora.

```
library(dplyr) #filter
library(bitops) # necesario con dplyr
library(RCurl) # getURL

while(1==1){
```

2. Uno de los parámetros a indicar en la URL es la fecha de inicio, es decir, desde que fecha se desea consultar los datos. Con la finalidad de obtener datos para todas las horas, la fecha de inicio deberá ser la fecha máxima almacenada hasta el momento en la estructura final para la Junta de Andalucía sumándole una hora.

```
  ########################################################################
  # Se construye la ruta local donde se encuentra la Estructura Final  #
  # y se carga la Estructura Final en myStoredGlobalData               #
  ########################################################################
  indicador<-83
  strGlobalDataPath<-"/myOpenData/path/data_global/datav2/"
  strGlobalStoredDataPath<-paste(strGlobalDataPath,indicador,"_myStoredDatav2.csv",sep="")
  myStoredGlobalData<-
read.csv(strGlobalStoredDataPath,header=TRUE,fill=FALSE,strip.white=TRUE)

  ########################################################################
  # Dentro de myStoredGlobalData se excluyen todos aquellos que no sean #
  # de la Red Hidrosur. El data frame resultado se almacena en          #
  # myStoredDataFilter                                                  #
  ########################################################################
  myStoredDataFilter<-filter(myStoredGlobalData,comaut==2 & ciudad==2)

  ########################################################################
  # Se obtiene la fecha máxima dentro de myStoredDataFilter            #
  # con la finalidad de usar esta fecha + 1 hora como la fecha inicio  #
  # para realizar una nueva consulta                                   #
  ########################################################################
  max_date<-max(as.POSIXct(myStoredDataFilter$fechal))
  str_maxDateStored<-max_date
  str_parameterDate<-str_maxDateStored+(60*60)

  ########################################################################
  # Se construye el parámetro a concatenar a la URL base               #
  ########################################################################
  str_finiPar<-paste("fechaini=",paste(format(str_parameterDate,"%Y-%m-
%d"),format(str_parameterDate,"%H"),sep="%20"),":00",sep="")
```

3. El otro parámetro es la fecha de fin. Esta fecha será la fecha actual del sistema, lo que

indicará que se quiere obtener desde la fecha de Inicio hasta los datos más recientes.

```
  ######################################################################
  # La fecha fin para la consulta será la fecha/hora actual (sistema)  #
  ######################################################################
  str_maxDateStored<-Sys.time()
  str_parameterDate<-str_maxDateStored

  ######################################################################
  # Se construye el parámetro a concatenar a la URL base               #
  ######################################################################
  str_ffinPar<-paste("fechafin=",paste(format(str_parameterDate,"%Y-%m-
%d"),format(str_parameterDate,"%H"),sep="%20"),":00",sep="")
```

4. Se construye la URL final en str_*url* con los parámetros ya previamente construidos. (No se debe especificar la estación, ya que la finalidad es descargar datos de todas las estaciones)

```
  ####################################################
  # URL base de los datos de la red Hidrosur         #
  ####################################################
  str_basePath<-"http://www.redhidrosurmedioambiente.es/webgis2/incphp/txt/datostemp.php?"
  str_estaPar<-"terminal=0"

  ######################################################
  # Se construye la URL concatenando a la URL base los #
  # parámetros que deseamos consultar                  #
  ######################################################
  str_url<-paste(str_basePath,paste(str_estaPar,str_finiPar,str_ffinPar,sep="&"),sep="?")
```

Anexo IX

Proceso: Integración datos de la Junta de Andalucía (Red Hidrosur)

Versión: R

Sub-proceso: Carga de Estructura Temporal-R

DETALLE DE IMPLEMENTACIÓN

La instrucción para implementar este sub-proceso es:

1. Para la descarga del fichero se usa la función *getURL* de la librería *Rcurl*. Además, la URL debe ser la que se tiene almacena en *str_url*. Y la estructura temporal en la que se ha cargado los datos descargados es *myCurrentData*.

```
  ########################################################
  # Se realiza la descarga del fichero y su contenido se #
  # almacena en myCurrentData                            #
  ########################################################
  myCurrentData<-read.csv(textConnection(getURL(str_url)), header = TRUE, stringsAsFactors =
FALSE, sep = ";")
```

Anexo X

Proceso: Integración datos de la Junta de Andalucía (Red Hidrosur)

Versión: R

Sub-proceso: Formateo-R

DETALLE DE IMPLEMENTACIÓN

Las instrucciones para implementar este sub-proceso es:

1. Antes de empezar este subproceso, se verifica que el fichero descargado contenga datos.

```
#########################################################
# Si el tamaño de los datos descargados es mayor que 0 #
#########################################################
if (nrow(myCurrentData)>0 )
{
```

2. Se añaden las columnas de Comunidad Autónoma (*comaut*), Ciudad (*ciudad*) e Indicador (*parame*).

```
  ####################################
  # 3 nuevas columnas a myCurrentData #
  # comaut : 2 (Andalucía)            #
  # ciudad : 2 (Ciudades Andalucía)   #
  # parame : indicador                #
  ####################################
  myCurrentData$comaut <- 2
  myCurrentData$ciudad <- 2
  myCurrentData$parame <- indicador
```

3. Se añaden las columnas *fechal* y *valorl* a partir de columnas que ya existen en *myCurrentData*.

```
  ##########################################################################
  # 2 nuevas columnas                                                      #
  # fechal : Se construye a partir de la columna 2 de myCurrentData (se    #
  #          formatea la fecha)                                            #
  # valorl : El valor de medición indicado en la columna 3 de myCurrentData  #
  #          (Se reemplaza "," por ".")                                    #
  ##########################################################################
  myCurrentData$fechal<-as.character(as.POSIXct( myCurrentData[,2 ], format =
"%d/%m/%y %H:%M"),format="%Y-%m-%d %H:%M:%S")
  myCurrentData$valorl<-as.numeric(gsub(",",".",myCurrentData[,3]))
```

4. Se añade la columna *estaci* que indicará la estación. Esta se construye a partir de la primera columna de *myCurrentData* que está compuesto de 3 campos y de los cuales solo nos interesa el primer campo que es el código de estación.

```
  ##########################################################################
  # La primera columna de myCurrenData tiene el siguiente formato          #
```

```
# [Código de Estación] - [Descripción de EStación] - [Código de Sensor]   #
# Ejemplo : 3 - EMBALSE DE CHARCO REDONDO (CA) - (003M02)                 #
# Solo interesa el Código de estación en una nueva columna estaci         #
###########################################################################
myCurrentData$estaci <- sapply(strsplit(myCurrentData[,1  ], " "), "[", 1)
```

5. Se eliminan las columnas originales de *myCurrentData* (se mantiene las columnas nuevas: comaut, ciudad, estaci, parame, fechal, valorl) y se reordena el data frame.

```
#########################################################
# Se eliminan las columnas originales de myCurrentData  #
# porque ya las tenemos en otro formato en otras nuevas #
# y reordenamos                                         #
#########################################################
myCurrentData<-myCurrentData[,-(1:4)]
names(myCurrentData) <- c("comaut","ciudad","estaci","parame","fechal","valorl")
```

Anexo XI

Proceso: Integración datos de la Junta de Andalucía (Red Hidrosur)

Versión: R

Sub-proceso: Inserción en Estructura Final-R

DETALLE DE IMPLEMENTACIÓN

Las instrucciones para implementar este sub-proceso es:

1. Se carga en *myStoredData* los datos que ya contiene la estructura final para el indicador.

```
#########################################################################
# Se carga en myStoredData los datos del indicador que se               #
# encuentran ya en el fichero de Estructura Final                       #
#########################################################################
str_dataPath<-"/myOpenData/path/data_JAL/datav2/"
str_storedDataPath<-paste(str_dataPath,indicador,"_myStoredDatav2.csv",sep="")

if (file.exists(str_storedDataPath))
{
  myStoredData<-read.csv(str_storedDataPath,header=TRUE,fill=FALSE,strip.white=TRUE)
}
```

2. Se añade en *myStoredData* los datos nuevos que se han descargado (*myCurrentData*).

```
#########################################################################
# Se añade a la Estructura Final (myStoredData) los datos de myCurrentData  #
#########################################################################
myStoredData<-rbind(myStoredData,myCurrentData)
```

3. En el fichero de la estructura final de datos se almacena *myStoredData* (*write.csv*) que ya contiene los nuevos datos.

```
#########################################################################
# myStoredData se guarda en el fichero de la Estructura Final de datos      #
#########################################################################
write.csv(myStoredData, file=str_storedDataPath,row.names=FALSE,quote=FALSE)
```

4. Se cierra la condición que verifica que el fichero descargado tenga datos, y finalmente se hace dormir el proceso para que empiece nuevamente dentro de una hora.

```
  } # Fin del If para verificar que myCurrentData tenga datos

  Sys.sleep(3600)

} # Fin del loop while
```

Estas son todas las instrucciones para integrar los datos de la Comunidad de Andalucía

dentro de la estructura final de datos.

Anexo XII

Proceso: Interface Gráfica

Versión: Java-Oracle-R

DETALLE DE IMPLEMENTACIÓN

Para detallar la implementación vamos a continuar con la numeración de instrucciones que nos quedamos en la sección de implementación del subproceso "Inserción en Estructura Final – Oracle" de la versión Java-Oracle del proceso de integración para el dataset de la Comunidad de Madrid. La implementación se basa en los siguientes pasos:

1. Se instancia un objeto de la clase *RscriptGenerator.*

```
RScriptGenerator rs = new RScriptGenerator("opendata","usuario","password");
```

2. Se calcula el día fecha del sistema para posteriormente pasarlo como parámetro para la generación del script

```
Date date = new Date();
Calendar cal = Calendar.getInstance();
cal.setTime(date);
day=cal.get(Calendar.DAY_OF_MONTH);
```

3. Se construye el script. Los parámetros enviados en este ejemplo son:

   Indicador: "Temperatura" (Código 83)

   Estación: "Pza. Castilla" (Código 50)

   Día: Día de la fecha del Sistema (generada en el paso anterior)

```
rs.build_Rscript(83, 50, day);
```

   El script R generado tendrá la siguiente apariencia:

```
library(RJDBC)
drv <-JDBC("oracle.jdbc.OracleDriver",
           classPath="/ruta/de/driver/ojdbc6.jar", " ")
con <- dbConnect(drv, "jdbc:oracle:thin:@localhost:1521/opendata",
           "usuario", "password")
y=dbGetQuery(con, "select hora, valor from t_datos where
           fk_parametro=83 and fk_estacion=50 and dia=15 order by hora")
jpeg(filename="/ruta/de/imagenes/plot_20160315021015.jpg")
plot(y[,2], type="o", col="blue", main="Temperatura por horas",
xlab="Horas", ylab="Grados Centigrados")
dev.off()
dbDisconnect(con)
```

4. Se ejecuta el script R generado:

```
      try {
      /*
             * Se ejecuta el Script R tal como si se ejecutará desde
             * línea de comando
       */
            Process child = Runtime.getRuntime().exec("/usr/bin/Rscript
"+rs.get_RScriptFileName());

            /*
             * Se espera la ejecución por un código de resultado
             */
            int code = child.waitFor();

            /*
             * Si el código devuelto es 0, el proceso se ejecutó correctamente
             * En caso contrario ha ocurrido un error.
             */
            if (code==0)
                  System.out.println ("El proceso R se ejecutó correctamente");

            else
                  System.out.println ("ERROR!!! al ejecutar el proceso R
"+rs.get_RScriptFileName());



      } catch (IOException e) {
            e.printStackTrace();
      } catch (InterruptedException e) {
            e.printStackTrace();
      }

/*
 * Esta última llave cierra el loop (while) que se abrió al inicio para hacer
 * dormir el proceso 3600000 milisegundos. Recordar que esta Versión es una
 * continuación de la Versión Java-Oracle del proceso de integración.
 */
}
```

Anexo XIII

Proceso: Interface Gráfica

Versión: JavaScript-Java-R

DETALLES DE IMPLEMENTACIÓN

Para detallar la implementación se detallará cada uno de los ficheros que son scripts mostrados en la
Tabla 34.

1. **Inicio.jsp:** En este script, se generarán cada uno de los 7 elementos del formulario. Para generar la información que se mostrará en cada combo, previamente se tiene que tener almacenada esta información en ficheros, es decir, la información de indicadores en el fichero parámetrosOD.csv (ver Figura 27) y la información de estaciones en el fichero estacionesOD.csv (ver Figura 28).

   Antes de empezar, es necesario mencionar que se van a usar librerías de Jquery para la generación de algunos campos, por lo tanto tenemos que hacer referencia a estas librerías:

```
<link rel="stylesheet" type="text/css" href="datetimepicker-
master|jquery.datetimepicker.css"/>
<link rel="stylesheet" type="text/css" href="timepicker\jquery-ui-timepicker-addon.css"/>
<link rel="stylesheet" media="all" type="text/css" href="jquery\jquery-ui.css" />

<script src="datetimepicker-master\jquery.js"></script>
<script src="datetimepicker-master\build\jquery.datetimepicker.full.min.js"></script>

<script src="jquery\jquery-1.11.1.min.js" ></script>
<script src="jquery\jquery-ui.min.js" ></script>

<script src="timepicker\jquery-ui-timepicker-addon.js"></script>
<script src="timepicker\i18n\jquery-ui-timepicker-addon-i18n.min.js"></script>
```

   A continuación se detallará el contenido de Inicio.jsp y su funcionamiento.

   1. Usamos JSP para usar la función *getStaticData* con el objetivo de obtener la información para posteriormente mostrar en el combo de indicadores y en el combo de estaciones.

```
<body>

<%

ArrayList<ArrayList<String>> al_listaParametros;
al_listaParametros = new ArrayList<ArrayList<String>>();

ArrayList<ArrayList<String>> al_listaEstaciones;
al_listaEstaciones = new ArrayList<ArrayList<String>>();
```

```
al_listaInd = GenericFunctions.getStaticData("/ruta/de/parametrosOD.csv", ",");
al_listaEst = GenericFunctions.getStaticData("/ruta/de/estacionesOD.csv", ",");

%>
```

2. Generamos el combo de indicadores y en el mostramos todos los indicadores obtenidos desde el fichero parametrosOD.csv

```
<a>Parametro :</a>
<select id="select_Parametro" onchange="fu_ActualizarEstacionFecha(this.value)">
<option value="0">Seleccionar Parametro</option>

<%
for (ArrayList<String> e : al_listaInd){
%>

<option value="<%=Integer.parseInt(e.get(0))%>"><%=e.get(1)%></option>

<%}%>

</select>
```

Observamos que se indica que se ejecute la función *fu_ActualizarEstacionFecha* cada vez que se cambie de selección en este combo. Esta función se detalla más a profundidad en la siguiente sección.

3. Se genera el combo de estaciones y en el mostramos todas las estaciones obtenidas desde el fichero estacionesOD.csv.

```
<div id="id_select_Estacion">
<a>Estacion :</a>
<select id="select_Estacion" onchange="fu_cambiar_Fechas(this.value)">
<option value="0">Seleccionar Estación</option>

<%
for (ArrayList<String> e : al_listaEstaciones){
%>

<option value="<%=Integer.parseInt(e.get(0))%>"><%=e.get(1)%></option>

<%}%>

</select>
</div>
```

Observamos que se indica que se ejecute la función *fu_cambiar_Fechas* cada vez que se cambie de selección en este combo. Esta función tiene se detalla más a profundidad en la siguiente sección.

4. Se generan los campos de fechas. Estos campos se generan como campos de texto, pero luego se indica que van a ser del tipo “dateTimePicker” de Jquery y que formarán una estructura Jquery “dateTimeRange”, esta estructura a su vez permite coordinar ambas fechas para que se puedan enviar como un Rango de fechas

```
<div id="id_Input_Fecha">
<!--Se declara los campos de fechas como si fuesen de texto-->
<a>Desde :</a>
<input type="text" id="id_Fecha_Desde"/>
<a>Hasta :</a>
<input type="text" id="id_Fecha_Hasta"/>

<br>
<script >
/*
 * Se indica que cada campo declarado previamente pase
 * a ser del tipo datetimepicker y luego formen un tipo
 * datetimerange de Jquery
 */
var startDateTextBox = $('#id_Fecha_Desde');
var endDateTextBox = $('#id_Fecha_Hasta');

$.timepicker.datetimeRange(
      startDateTextBox,
      endDateTextBox,
      {
            minInterval: (1000*60*60),
            dateFormat: 'dd-mm-yy',
            timeFormat: 'HH:mm',
            start: {},
            end: {}
      }
);
</script>
</div>
```

5. Se genera el elemento CheckBox que indica si se sobreponen los gráficos o no

```
<input type="checkbox" id="id_mantener" value="Mantener"
onchange="fu_Verificar_Fechas_Parametro()">Mantener Imagenes
```

Observamos que cuando el usuario modifique este elemento, se ejecutará la función *fu_Verificar_Fechas_Parametro*. Esta función tiene se detalla más a profundidad en la siguiente sección.

6. Se genera el botón con el que se generará el gráfico.

```
<br>
<button id="id_boton" onclick="fu_Generar_Imagen()">Generar</button>
<br>
```

Observamos que cuando se pulsa sobre el botón, se ejecuta la función *fu_Generar_Imagen*. Esta función tiene se detalla más a profundidad en la siguiente sección.

7. Se genera el espacio/elemento sobre el cual se mostrarán los gráficos generados. (Al inicio se mostrará vacío pero luego se actualizará por medio de Ajax, al igual que otros elementos antes mencionados)

```
<div id="Imagen"></div>
```

8. Por último, se deshabilita todos los elementos, excepto el selector del indicador. Esto es con la finalidad de que el usuario siempre escoja primero el indicador y luego los otros parámetros.

```
<script>
  $('#select_Estacion').prop('disabled', 'disabled');
  document.getElementById("id_Fecha_Desde").disabled = true;
  document.getElementById("id_Fecha_Hasta").disabled = true;
  document.getElementById("id_boton").disabled = true;
</script>

</body>
```

La Tabla 37 muestra las funciones que se usa en el script Inicio.jsp.

Tabla 37. Funciones genéricas para la Interface Gráfica (JavaScript-Java-R)

Nº	Método	Descripción
1	**fu_ActualizarEstacionFecha**	Esta función valida si es que hay datos para el indicador especificado. El objetivo de esta función es actualizar el combo de la estación y los campos de las fechas para mostrar los valores disponibles en base al Indicador seleccionado en el combo de Indicadores. Para conseguir esto, ejecuta el jsp: *AlertNoFile.jsp* con Ajax.
2	**fu_cambiar_Estacion**	Esta función modifica el combo de estaciones para mostrar solo aquellas estaciones que midan el indicador escogido en el combo de indicadores. Es decir el combo de estaciones esta anidado al combo de indicadores. Esta actualización que se hace sobre el combo de estaciones se realiza con el script *ActualizarEstacionCombo.jsp*. Este script lo detallaremos más adelante. Este es un ejemplo de dinamismo que se consigue mediante Ajax.
3	**fu_cambiar_Fechas**	Esta función modifica los campos de fechas para mostrar valores por defecto en : El campo de “Desde”: La fecha desde la cual se empezó a recibir datos para ese indicador en cualquier estación o para ese indicador en una

		Estación indicada como parámetro. El campo "Hasta": La última fecha para la cual se ha recibido datos para ese indicador en cualquier estación o para ese indicador en una estación indicada como parámetro. Es decir el combo de estaciones esta anidado al combo de Indicadores y/o estaciones. Esta actualización que se hace sobre los campos de fechas se realiza con el script *ActualizarFechasInput.jsp*. Este script lo detallaremos más adelante. Este es otro ejemplo de dinamismo que se consigue mediante Ajax.
4	**fu_Verificar_Fechas_Parametro**	Esta función está bastante asociada a lo que hace el elemento CheckBox "Mantener Imagen". Como se ha mencionado previamente, este CheckBox permite sobreponer los gráficos para poder realizar comparaciones. Pero **hacer una comparación implica que se esté tratando del mismo indicador, así como también sobre el mismo intervalo de tiempo**. Esta función realiza estas validaciones.
5	**fu_Generar_Imagen**	Esta función valida que se hayan especificado todos los campos. Y luego genera el gráfico usando Ajax.

2. **ActualizarEstacionCombo.jsp:** Este script jsp tiene por finalidad extraer las estaciones correspondientes a un indicador enviado como parámetro. Este script ha sido invocado mediante Ajax y lo que genere este script será reemplazado dentro del elemento con ***id="id_Select_Estacion"*** en la interface gráfica.

3. **ActualizarFechasInput.jsp:** Este script jsp tiene por finalidad extraer las fechas correspondientes a un indicador y/o estación enviados como parámetro. Este script ha sido invocado mediante Ajax y lo que genere este script será reemplazado dentro del elemento con ***id="id_Input_Fecha"*** en la interface gráfica.

4. **AlertNoFile.jsp:** Este script tiene como finalidad validar si para un indicador especificado en la interface gráfica existen datos en la estructura de datos.

5. **ActualizarImagen.jsp:** Este script es el encargado de llamar al script R: generarPlots.r mediante el cual se genera el gráfico y una leyenda del mismo. Otra vez se ha usado Ajax para invocar a este script.

6. **GenericFunctions.r:** Este script R será de utilidad también para detallar otras versiones, ya que contiene funciones desarrolladas en R de carácter genérico. Las funciones de este script se muestran en la Tabla 38.

Tabla 38. Funciones genéricas del Script GenericFunctions.r

Nº	**Método**	**Descripción**
1	getUniqueValues	Devuelve valores únicos de una columna de un data frame
2	loadStaticData	Recoge la información de ficheros estáticos como lo son los indicadores y estaciones y los almacena en un data frame global.
3	getOpenDataDates	Esta función devuelve la fecha máxima o mínima en base al indicar recibido como parámetro.
4	genPlot	Esta función es la que genera el gráfico leyendo los parámetros desde el fichero ***mantener.tmp*** y guardando el nombre del gráfico en el fichero data.tmp para que pueda ser mostrado posteriormente por la interface gráfica.

7. **WebInterface.r**: La finalidad de este script es crear ficheros temporales de datos los cuales puedan servir de comunicación entre R y Java ya que en muchos scripts jsp (desde código Java embebido) se ejecuta scripts R enviando parámetros. Sin embargo, no se encontró la manera de como R devuelva una respuesta que se pueda procesar en código Java.

   Este script crea los ficheros que sirvan de respuesta desde R hacia Java. Los ficheros temporales a crear son dataEst.tmp y dataDte.tmp. Dependiendo del caso, se puede crear

solo el dataEst.tmp o se pueden crear ambos. Según la Tabla 34, el fichero **dataEst.tmp** almacena información de las estaciones a mostrarse en el combo de Estaciones una vez se especifique un indicador, mientras que el fichero **dataDte.tmp** almacena información de las fechas a mostrarse en los campos de fechas del formulario una vez que se especifique un indicador o un indicador y estación.

Primero se importan las funciones del script ***GenericFunctions.r*** y luego recogen los parámetros. El comportamiento es distinto para cuando se recibe solo el indicador y para cuando se recibe el indicador y la estación. En el primer caso, cuando se recibe solo el indicador, Se va a buscar tanto las estaciones disponibles como el rango de fechas en los que existen datos para ese indicador. Estos valores son calculados con la finalidad de filtrar valores en los combos de las estaciones y los campos de las fechas.

En el segundo caso, cuando se recibe el indicador y la estación, se va a calcular solo el rango de fechas en los que existen datos para el indicador y estación seleccionados. Estos valores son calculados con la finalidad de filtrar valores en los campos de las fechas. Estos filtros son útiles para limitar las opciones de selección posibles al usuario con el fin de evitar que este seleccione valores inconsistentes.

8. **GenerarPlots.r:** Este script recoge el indicador que viene en el parámetro y luego invoca a la función ***genPlot*** del script ***GenericFunctions.r***

Anexo XIV

Proceso: Interface Gráfica

Versión: R

DETALLES DE IMPLEMENTACIÓN

Esta versión de la implementación de la interface gráfica hace uso de los dos scripts siguientes:

➢ *ui.r*

En este script se definen cada uno de los elementos de la interface gráfica.

```
#########################################################
# Se obtiene los datos de las regiones integradas     #
#########################################################

strDataPath<-"/myOpenData/path/data_global/"

strFilePath<-paste(strDataPath,"comAut.csv",sep="")
myComAut<-read.csv(strFilePath,header=TRUE,stringsAsFactors=FALSE)
myComAut<-setNames(myComAut[,1],myComAut[,2])


#########################################################
# En la siguiente función se definen los elementos    #
# de la GUI                                           #
#########################################################
shinyUI(pageWithSidebar(

  ############################
  # Se define un título    #
  ############################
  headerPanel("Datos Ambientales en Territorio Español"),

  ############################################################
  # Se define un panel en el cual se ubiquen los campos    #
  # para que el usuario indique los parámetros             #
  ############################################################
  sidebarPanel(

    ############################################################
    # Se genera un Combo que permita la selección de la región #
    ############################################################
    selectInput("region", "Region:",myComAut),

    ############################################################
    # Se genera un campo para elegir el indicador             #
    # Posteriormente este elemento se convertirá en un Combo   #
    # en el cual se muestren los indicadores disponibles para  #
    # la región seleccionada                                   #
    ############################################################
    uiOutput("elegir_indicador"),

    ############################################################
    # Se genera un campo para elegir la estación              #
    # Posteriormente este elemento se convertirá en un Combo   #
    # en el cual se muestren las estaciones disponibles para   #
    # la región/indicador seleccionados                        #
    ############################################################
    uiOutput("elegir_estacion"),

    ############################################################
```

```
    # Se genera un campo para indicar la fecha de Inicio       #
    # este campo será brindado de dinamismo                    #
    ############################################################
    uiOutput("elegir_fechaDesde"),
    ############################################################
    # Se genera un elemento slider que servirá para indicar la #
    # hora de Inicio                                           #
    ############################################################
    sliderInput("horaDesde","Hora Desde:",min = 0,max = 23,value=0),

    ############################################################
    # Lo mismo para la fecha/hora de Fin                       #
    ############################################################
    uiOutput("elegir_fechaHasta"),
    sliderInput("horaHasta","Hora Hasta:",min = 0,max = 23,value=23)
  ),

  ############################################################
  # Se define un panel en el cual se ubique el gráfico con #
  # un título                                                #
  ############################################################
  mainPanel(
    ############################################################
    # Se ubica un título (elemento caption) y una imagen       #
    # (elemento mpgPlot)                                       #
    ############################################################
    h5(textOutput("caption")),
    plotOutput("mpgPlot")
  )
))
```

➢ *server.r*

Por regla general este script define una lógica de servidor, del cual se consultan datos. La función de este script es generar dinamismo entre los campos del formulario de la interface gráfica. A continuación detallamos como genera dinamismo para cada uno de los campos de la interface gráfica.

```
##############################
# Se añade la librería dplyr #
##############################
library(dplyr)

#################################################################################
# Se define la función ShinyServer                                              #
# Por medio de esta función es posible incluso dar dinamismo al formulario  #
#################################################################################
shinyServer(function(input, output) {
```

1. Se añade dinamismo al título del gráfico

```
  ####################################################################
  # Se define un título para el gráfico (elemento de salida "caption") #
  ####################################################################
  formulaText <- reactive({
    paste("Indicador :", input$indicador,"\nEstacion :",input$estacion,"\nDe :
",input$fechaDesde,"\nA :",input$fechaHasta)
  })
  output$caption <- renderText({
    formulaText()
  })
```

2. Se añade dinamismo al combo de indicadores

```
  ##########################################################
  # Se define el Combo de entrada para indicadores         #
  # la sentencia renderUI, indica que tiene comportamiento #
  # dinámico cada vez que se modifique algún parámetro     #
  ##########################################################
  output$elegir_indicador <- renderUI({

    ##########################################################
    # Se recoge el parámetro de entrada región para generar   #
    # buscar los indicadores asociados a esa región          #
    ##########################################################
    if(is.null(input$region))
      return()

    myIndicadores<-
read.csv("/myOpenData/path/data_global/datav2/parametrosOD.csv",header=TRUE,fill=FALSE,strip.w
hite=TRUE)
    myIndicadores<-select(filter(myIndicadores,comaut==input$region),parame,descripcion)
    myIndicadores<-setNames(myIndicadores[,1],myIndicadores[,2])

    ##########################################################
    # Con los indicadores obtenidos en myIndicadores se      #
    # genera un elemento Combo para Indicadores              #
    ##########################################################
    selectInput("indicador", "Indicador", myIndicadores)

  })
```

3. Se añade dinamismo al combo de estaciones

```
  ##########################################################
  # Se define el Combo de entrada para estaciones          #
  # la sentencia renderUI, indica que tiene comportamiento #
  # dinámico cada vez que se modifique algún parámetro     #
  ##########################################################
  output$elegir_estacion <- renderUI({

    ##########################################################
    # Se recogen los parámetros de entrada región e indicador #
    # para generar las estaciones asociadas a esos 2          #
    # parametrons                                              #
    ##########################################################
    if(is.null(input$indicador) || is.null(input$region))
      return()


#############################################################################################
# En base al indicador seleccionado se selecciona el dataFrame con los datos                #
# correspondientes al indicador                                                             #
#############################################################################################
    sNameParFile<-
paste("/myOpenData/path/data_global/datav2/",input$indicador,"_myStoredDatav2.csv",sep="")
    myStoredData<-read.csv(sNameParFile,header=TRUE,fill=FALSE,strip.white=TRUE)
    myStoredData<-myStoredData[myStoredData[,1]==input$region,]

    ##############################################
    # De ese dataFrame se descartan los repetidos #
    ##############################################
    dfUniqueCodes<-distinct(select(myStoredData,estaci))
    names(dfUniqueCodes)<-c("estacion")

    ##########################################################
    # Se crea un dataFrame  con todos las estaciones         #
    ##########################################################
    myEstaciones<-
read.csv(file="/myOpenData/path/data_global/datav2/estacionesOD.csv",header=FALSE,fill=FALSE,s
trip.white=TRUE)
    myEstaciones<-myEstaciones[myEstaciones[,1]==input$region,]
    names(myEstaciones)<-c("comaut","estacion","decripcion")
```

```
    ##################################################################
    # Se hace un merge (inner join) entre ambos dataframes para tener #
    # un único dataframe con la forma (código,descripción) únicos    #
    ##################################################################
    dfMerge<-merge(dfUniqueCodes,myEstaciones,by="estacion")[,c(1,3)]

    myEstaciones<-setNames(dfMerge[,1],dfMerge[,2])

    ##########################################################
    # Con los estaciones obtenidos en myEstaciones se genera #
    # un elemento Combo para Indicadores                     #
    ##########################################################
    selectInput("estacion", "Estaciones:",myEstaciones)

  })
```

4. Se añade dinamismo al campo de “Fecha de Inicio”

```
  ##########################################################
  # Se define el campo de entrada para la fecha Inicio     #
  # la sentencia renderUI, indica que tiene comportamiento #
  # dinámico cada vez que se modifique algún parámetro     #
  ##########################################################
  output$elegir_fechaDesde <- renderUI({

    ##########################################################
    # Se obtiene la fecha mínima para ese indicador/región  #
    ##########################################################
    sNameParFile<-
paste("/myOpenData/path/data_global/",input$indicador,"_myStoredData.csv",sep="")
    myStoredData<-read.csv(sNameParFile,header=TRUE,fill=FALSE,strip.white=TRUE)
    myStoredData<-myStoredData[myStoredData[,1]==input$region,]
    fechaMinima<-min(myStoredData$fechal)

    ##########################################################
    # Con la fecha mínima se genera el elemento de fecha de  #
    # de inicio en el cual establecemos la fecha mínima como #
    # valor por defecto                                      #
    ##########################################################
    dateInput("fechaDesde", "Fecha Desde:", min = fechaMinima )

  })
```

5. Se añade dinamismo al campo de “*Fecha de Fin”*

```
  ##########################################################
  # Se define el campo de entrada para la fecha Fin        #
  # la sentencia renderUI, indica que tiene comportamiento #
  # dinámico cada vez que se modifique algún parámetro     #
  ##########################################################
  output$elegir_fechaHasta <- renderUI({

    ##########################################################
    # Se obtiene la fecha mínima para ese indicador/región  #
    ##########################################################
    sNameParFile<-
paste("/myOpenData/path/data_global/",input$indicador,"_myStoredData.csv",sep="")
    myStoredData<-read.csv(sNameParFile,header=TRUE,fill=FALSE,strip.white=TRUE)
    myStoredData<-myStoredData[myStoredData[,1]==input$region,]
    fechaMaxima<-max(myStoredData$fechal)

    ##########################################################
    # Con la fecha mínima se genera el elemento de fecha de  #
    # de inicio en el cual establecemos la fecha mínima como #
    # valor por defecto                                      #
    ##########################################################
    dateInput("fechaHasta", "Fecha Hasta:", max = fechaMaxima, min =
as.character(input$fechaDesde))

  })
```

6. Se añade dinamismo en la generación del gráfico.

```
  ############################################################
  # Se define la imagen que se va a generar                  #
  # la sentencia renderPlot indica que tiene comportamiento  #
  # dinámico cada vez que se modifique algún parámetro       #
  ############################################################
  output$mpgPlot <- renderPlot({


    ############################################################
    # Se carga el data frame del indicador recibido como input #
    ############################################################
    sNameParFile<-
paste("/myOpenData/path/data_global/datav2/",input$indicador,"_myStoredDatav2.csv",sep="")

    myStoredData<-read.csv(sNameParFile,header=TRUE,fill=FALSE,strip.white=TRUE)

    desde<-as.character(paste(input$fechaDesde,paste(input$horaDesde,":00:00",sep=""),sep="
"))
    hasta<-as.character(paste(input$fechaHasta,paste(input$horaHasta,":00:00",sep=""),sep="
"))

    ############################################################
    # Se descarta los datos que no cumplen con los parámetros  #
    # recibidos como input                                     #
    ############################################################
    y<-myStoredData[myStoredData[,1]==input$region & myStoredData[,3]==input$estacion &
as.POSIXct(myStoredData[,5],format="%Y-%m-%d %H:%M:%S")>=as.POSIXct(desde) &
as.POSIXct(myStoredData[,5],format="%Y-%m-%d %H:%M:%S")<=as.POSIXct(hasta),]


    ############################################################
    # Se cargan los datos estáticos para obtener descripciones #
    # del indicador como de la estación. Todo esto tiene por   #
    # finalidad construir un encabezado al gráfico             #
    ############################################################
    cStaticData<-
read.csv(file="/myOpenData/path/data_global/datav2/parametrosOD.csv",header=TRUE,fill=FALSE,st
rip.white=TRUE)
    cStaticData<-cStaticData[cStaticData[,1]==input$region &
as.integer(cStaticData[,2])==as.integer(input$indicador),]
    cParametro<-cStaticData[1,3]

    cStaticData<-
read.csv(file="/myOpenData/path/data_global/datav2/estacionesOD.csv",header=FALSE,fill=FALSE,s
trip.white=TRUE)
    cStaticData<-cStaticData[cStaticData[,1]==input$region & cStaticData[,2]==input$estacion,]
    cEstacion<-cStaticData[1,3]

    cTitulo<-paste(cParametro,"en",cEstacion,sep=" ")

    ################################################################
    # En el conjunto de datos sobre los cuales generar el grafico, #
    # se formatea la fecha en un formato legible para el usuario   #
    # en el gráfico. Luego descartamos todas las columnas excepto  #
    # la fecha y el valor                                          #
    ################################################################
    y$fechal<-as.POSIXct(y$fechal)
    y<-select(y,fechal,valorl)

    ###############################################
    # Finalmente se genera el gráfico             #
    ###############################################
    plot(y[,1],y[,2], type="o", col="blue", main=cTitulo, xlab="Horas", ylab=cParametro)

  })
})
```